

MATLAB应用技术

精通MATLAB

MATLAB APPLICATION

葛哲学 编著

飞思科技产品研发中心 监制

国内实力MATLAB专家多年经验积累

内容更全面：

从基础知识到高级功能，涵盖MATLAB的最主流技术

知识更精到：

注重知识的概括和凝练，并对MATLAB的高级应用进行深入探讨

案例更典型：

大量实际案例引领读者系统掌握MATLAB，更好地解决实际工程问题



電子工業出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

http://www.phei.com.cn

目　录

第 1 章　MATLAB 基础

本章的主要目的是对 MATLAB 的基本知识做全面介绍，向读者展示 MATLAB 软件的特点以及它的强大功能，将读者引入 MATLAB 的殿堂，并给读者学习、使用 MATLAB 以方法性的指导。

本章主要内容：

- MATLAB 简介
- MATLAB 软件安装、界面和帮助
- MATLAB 示例
- MATLAB 学习技巧

1.1　MATLAB 简介

MATLAB 是由 MathWorks 公司开发的一种主要用于数值计算及可视化图形处理的工程语言，是当今最优秀的科技应用软件之一。它将数值分析、矩阵运算、图形图像处理、信号处理和仿真等诸多强大的功能集成在较易使用的交互式计算机环境之中，为科学研究、工程应用提供了一种功能强、效率高的编程工具。它拥有强大的科学计算与可视化功能、简单易用、开放式可扩展环境，特别是所附带的 30 多种面向不同领域的工具箱支持，使得它在许多科学领域中成为计算机辅助设计和分析、算法研究和应用开发的基本工具和首选平台。

MATLAB 语言被通俗地称为演算纸式科学算法语言，在控制、通信、信号处理及科学计算等领域中得到了广泛的应用，已经被认可为能够有效提高工作效率、改善设计手段的工具软件。

1.1.1　MATLAB 发展史

MATLAB 名字由 Matrix（矩阵）和 Laboratory（实验室）两词的前 3 个字母组合而成。20 世纪 70 年代后期时任美国新墨西哥大学计算机系主任的 Cleve Moler 博士讲授线性代数课程，发现应用其他高级编程语言极为不方便，于是 Cleve Moler 博士和他的同事构思并为学生设计了一组调用 LINPACK 和 EISPACK 库程序的“通俗易用”的接口，这就是用 Fortran 编写的萌芽状态的 MATLAB。以后几年，MATLAB 作为免费软件在大学里被广泛使用，深受大学生的喜爱。

1984 年，John Little、Cleve Moler 和 Steve Bangert 合作成立了 MathWorks 公司，专门从事 MATLAB 软件的开发，并把 MATLAB 正式推向市场。从那时起，MATLAB 的内

核采用C语言编写，而且除原有的数值计算能力外，还新增了数据图视功能。1993年，MathWorks公司推出MATLAB 4.0版本；1995年，MathWorks公司推出MATLAB 4.2C版（For Win3.x）。4.x版在继承和发展其原有的数值计算和图形可视能力的同时，增加了一些功能：①推出Simulink；②开发出基于Word处理平台的Notebook；③推出符号计算工具包；④开发了与外部进行直接数据交换的组件，打通了MATLAB进行实时数据分析、处理和硬件开发的通路。1997年，MathWorks公司推出MATLAB 5.0；2000年10月推出了MATLAB 6.0；2002年8月，推出了MATLAB 6.5，从此MATLAB拥有了强大的、成系列的交互式界面。2004年7月，又进一步发展到了MATLAB 7.0，在MATLAB 7.0中，仿真模块发展到了Simulink 6.0。

MATLAB R系列是从2006年开始发布的，MathWorks公司在技术层面上实现了一次飞跃。从此以后产品发布模式也将改变，将在每年的3月和9月进行两次产品发布，版本的命名方式为“R＋年份＋代码”，对应上下半年的代码分别为a和b。每一次发布都会包含所有的产品模块，如产品的new feature、bug fixes和新产品模块的推出。MATLAB R2007a是MathWorks公司2007年3月份推出的最新产品。

1.1.2 MATLAB软件主要特点

MATLAB集计算、可视化及编程于一身。在MATLAB中，无论是问题的提出还是结果的表达都采用我们习惯的数学描述方法，而不需要用传统的编程语言进行前后处理。这一特点使MATLAB成为了数学分析、算法开发及应用程序开发的良好环境。MATLAB是MathWorks产品家族中所有产品的基础。MATLAB的主要特点如下。

1）强大的科学计算功能

MATLAB拥有500多种数学、统计及工程函数，可使用户立刻实现所需的强大的数学计算功能。由各领域的专家学者们开发的数值计算程序，使用了安全、成熟、可靠的算法，从而保证了最大的运算速度和可靠的结果。

2）先进的可视化工具

MATLAB提供功能强大的、交互式的二维和三维绘图功能。可创建富有表现力的彩色图形。可视化工具包括：曲面渲染（Surface Rendering）、线框图，伪彩图、光源，三维等高线图、图像显示、动画、体积可视化等。

MATLAB提供了Handle Graphics图形机制。使用该机制可对图形进行灵活的控制。使用GUIDE工具，我们可以方便地使用Handle Graphics创建自己的图形用户界面。

3）直观灵活的语言

MATLAB不仅仅是一套打包好的函数库，同时也是一种高级的、面向对象的编程语言。使用MATLAB可卓有成效地开发自己的程序。MATLAB自身的许多函数，实际上也包括所有的工具箱函数，都是用M文件实现的。

4）开放性、可扩展性强

M文件是可见的MATLAB程序，所以我们可以查看源代码。开放的系统设计使我们能够检查算法的正确性，修改已存在的函数，或者加入自己的新部件。

5）众多面向领域应用的工具箱和模块集

MATLAB 的工具箱加强了对工程及科学中特殊应用的支持。工具箱和 MATLAB 一样是完全用户化的，可扩展性强。将某个或某几个工具箱与 MATLAB 联合使用，可以得到一个功能强大的计算组合包，满足用户的特殊要求。

1.1.3 MATLAB 软件共生产品

MathWorks 公司开发的产品是一个非常庞大的系统家族，它主要包括 MATLAB 产品家族、Simulink 产品家族和二者的链接产品，如图 1-1 所示。MATLAB 是整个体系的基石，它是一个语言编程型（M 语言）开发平台，提供了体系中其他工具所需要的集成环境（比如 M 语言的解释器）。MATLAB 产品体系的演化历程中最重要的一个体系变更是引入了 Simulink，用来对动态系统建模仿真，其框图化的设计方式和良好的交互性，对工程人员本身的计算机操作与编程的熟练程度的要求降到了最低，工程人员可以把更多的精力放到理论和技术的创新上去。针对控制逻辑的开发、协议栈的仿真等要求，MathWorks 公司在 Simulink 平台上还提供了用于描述复杂事件驱动系统的逻辑行为的建模仿真工具——Stateflow，通过 Stateflow，用户可以用图形化的方式描述事件驱动系统的逻辑行为，并无缝地结合到 Simulink 的动态系统仿真中。

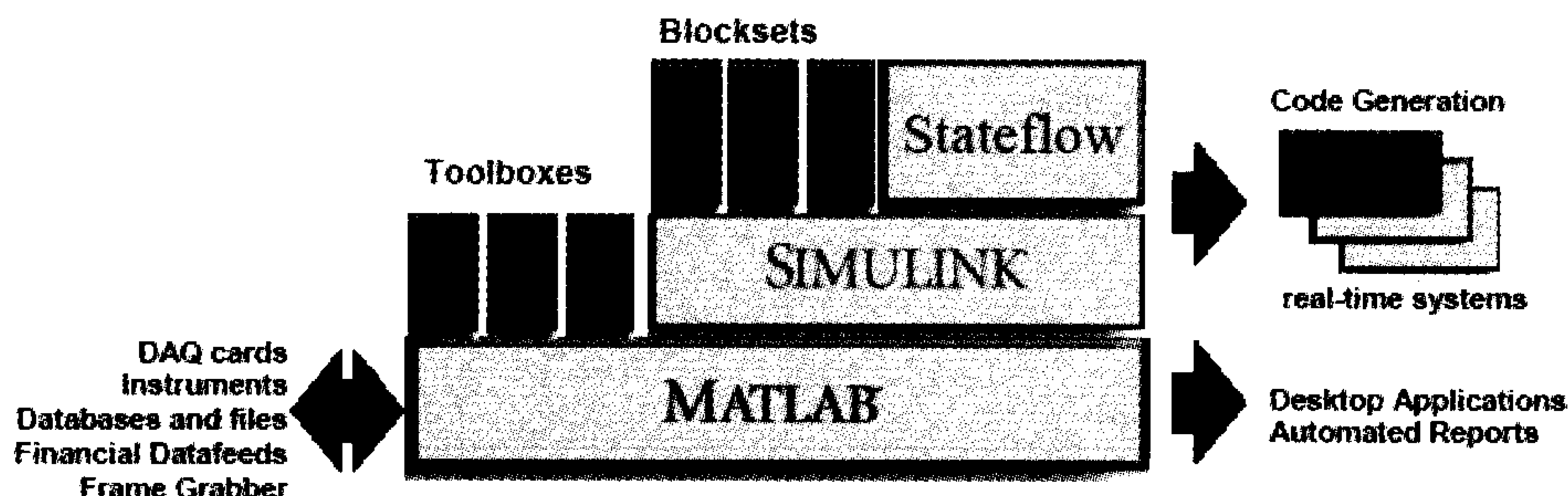


图 1-1　MathWorks 的产品体系

在 MATLAB/Simulink 基础环境之上，MathWorks 公司为用户提供了丰富的扩展资源，这就是大量的工具箱（Toolbox）和模块集（Blockset）。从 1985 年推出第一个版本以后的近 20 年发展过程中，MATLAB 已经从单纯的 Fortran 数学函数库演变为多学科、多领域的函数包及模块库的提供者。用户在这样的平台上进行系统设计开发就相当于已经站在了巨人的肩膀上，众多行业中的专家、精英们的智慧结晶可以信手拈来。随着 MATLAB 的不断扩充，工具箱将会越来越多。而工具箱实际上就是在 MATLAB 系统上开发的一组实用的 M 文件函数命令或者是 Simulink 仿真模型。因此，只要用户有兴趣和要求，自己也可以开发特殊用途的工具箱。如表 1-1 所示列出了 MATLAB 所有的产品模块，分别对应于不同领域的工具箱和 Simulink 模块集。

表 1-1　MATLAB 所有的产品模块

类　别	模 块 名 称	类　别	模 块 名 称
基础工具	MATLAB MATLAB report generator Simulink Simulink performance tool Simulink report generator Stateflow Stateflow Coder Real-Time Workshop	控制	Control System Fuzzy Logic Fixed-Point Blockset System Identification LMI Control Model Predictive Mu-Synthesis Nonlinear Control Design Blockset Robust Control
数学 & 金融	Curve fitting Database Toolbox Financial Derivatives Datafeed Toolbox Extended Symbolic Math Financial Financial Time Series Fixed-Income Toolbox GARCH Toolbox Optimization Partial Differential Equation Symbolic Math Spline Statistics Neural Network Bioinformatrics Toolbox	信号处理/图像处理/通信系统开发	CDMA Reference Blockset Communications Blockset Communications Toolbox SPC Blockset Image Acquisition Toolbox Signal Processing Toolbox Image Processing Toolbox Filter Design Toolbox Wavelet Toolbox Link for ModelSim
数学 & 金融	Curve fitting Database Toolbox Financial Derivatives Datafeed Toolbox Extended Symbolic Math Financial Financial Time Series Fixed-Income Toolbox GARCH Toolbox Optimization Partial Differential Equation Symbolic Math Spline Statistics Neural Network Bioinformatrics Toolbox	测试测量	Data Acquisition Toolbox Instrument Control

（续表）

类　别	模块名称	类　别	模块名称
实时目标系统	Real-Time Workshop Embedded Coder EmTrg for Infineon C166 Microcontrollers MATLABLink for Code Composer Studio Embedded Target for Motorola HC12 Embedded Target for Motorola MPC555 Embedded Target for OSEK/VDX Embedded Target for TI C6000 DSP Real-Time Windows Target xPC Target Embedded Option xPC Target Embedded Target for TI C2000 DSP	应用接口	MATLABCompiler MATLABCOM Builder MATLABExcel Builder Excel Link MATLABRuntime Server MATLABWeb Server
其他	Aerospace Blockset Model-Based Calibration Toolbox Mapping SimMechanics SimPowerSystems Virtual Reality Toolbox Dials & Gauges		

这些工具箱和模块的详细信息可以参见 MATLAB 的联机帮助。在本书第 7 章将示例性地介绍一些常用的工具箱。

1.1.4 MATLAB 软件组成

MATLAB 作为 MathWorks 产品家族的核心，它主要由五大部分组成，分别为 MATLAB 语言（the MATLAB Language）、MATLAB 工作环境（the MATLAB Working Environment）、MATLAB 数学函数库（the MATLAB Math Library）、图形句柄系统（Handle Graphics）和 MATLAB 应用程序借口（the MATLAB Application Interface）。下面对它们分别进行介绍。

1）MATLAB 语言

MATLAB 语言是一种以矩阵（Matrix）和阵列（Array）为基本编程单元的，拥有完整的控制语句、数据结构、函数编写与调用格式和输入输出功能的具有面向对象程序设计特征的高级程序语言。读者不但可以利用它方便快捷地完成小规模的算法验证、程序开发和调试工作，而且可以使用它进行大规模、高效的复杂应用程序设计。

2）MATLAB 工作环境

MATLAB 工作环境简而言之就是一系列实用工具的集合，它不但包括了各种操作工作空间中变量的工具和管理数据输入输出的方法，而且包括了开发调试 M 文件和 MATLAB 应用程序的集成环境，使用起来极为方便。当用户在 Windows NT 系统下启动 MATLAB 后，将会出现如图 1-2 所示的命令窗口（the Command Window），这是用户同 MATLAB 工作环境交互的主要窗口，在命令提示符“>>”下，用户可以输入各种相关命令，来完成所希望的操作。

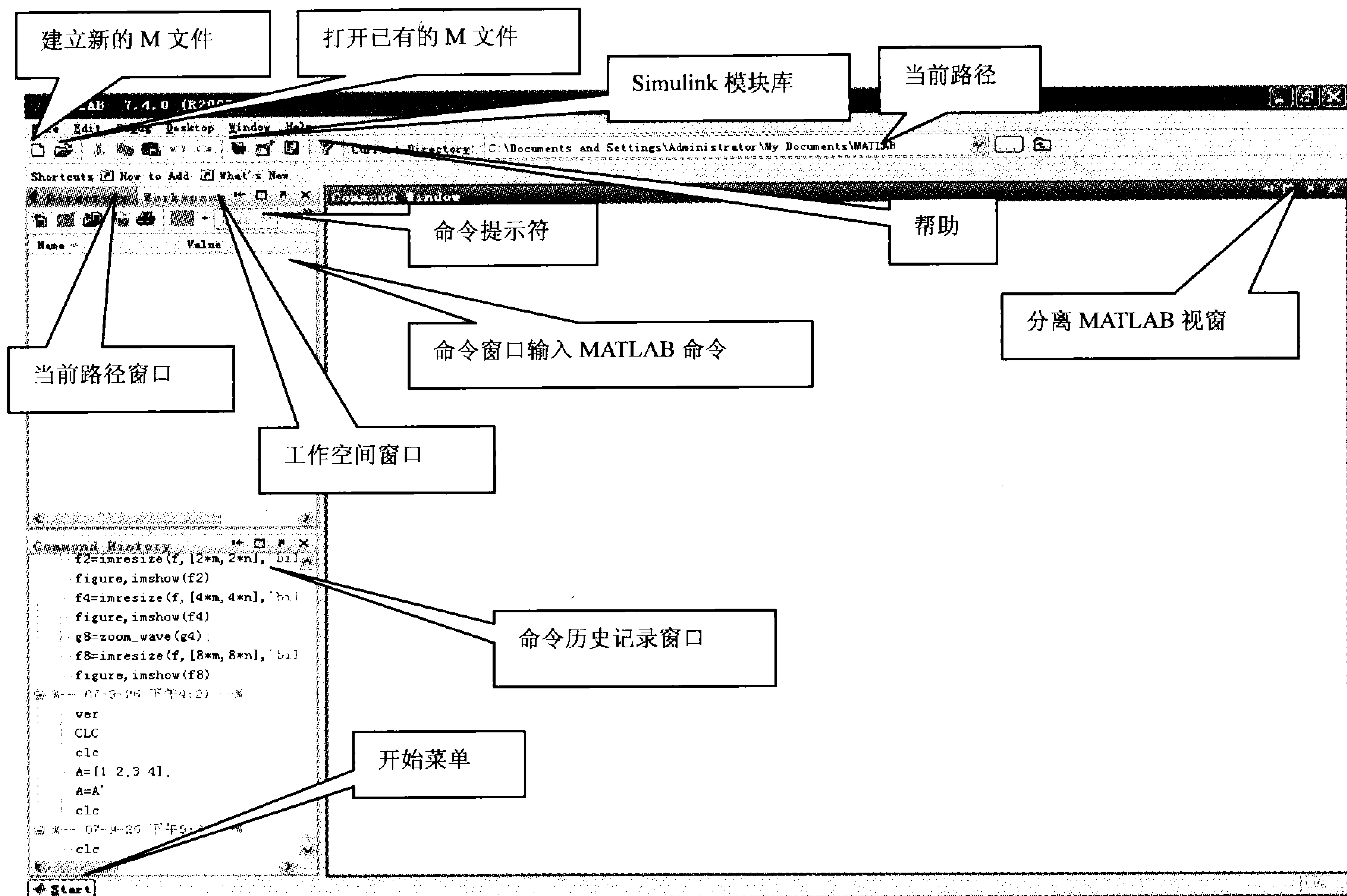


图 1-2 “MATLAB”命令窗口

在命令窗口中，用户除了可以在命令提示符下输入命令执行操作外，还可以通过菜单和工具栏执行多种任务，如图 1-3 所示，通过【建立新的 M 文件】按钮和【打开已有的 M 文件】按钮可以开启 M-edit 编辑调试器，如图 1-3 所示，这是一个功能非常完善的编辑调试环境；通过当前路径窗口，可以查看当前工作路径中各变量的类型和内容；通过工作空间窗口，用户可以查看通过 MATLAB 命令所操作的文件和结果的内容以及类型；通过【帮助】按钮，可以打开帮助窗口，让用户查找在线帮助；通过命令历史记录窗口，用户可以查看过去进行的 MATLAB 操作；通过【Simulink 模块库】按钮，可以打开 Simulink 模块库，让用户向自己的模型中添加新的模块。总之，MATLAB 工作环境是一个功能异常强大的工具集合，几乎可以令用户完成所有的操作，并且简单易用。

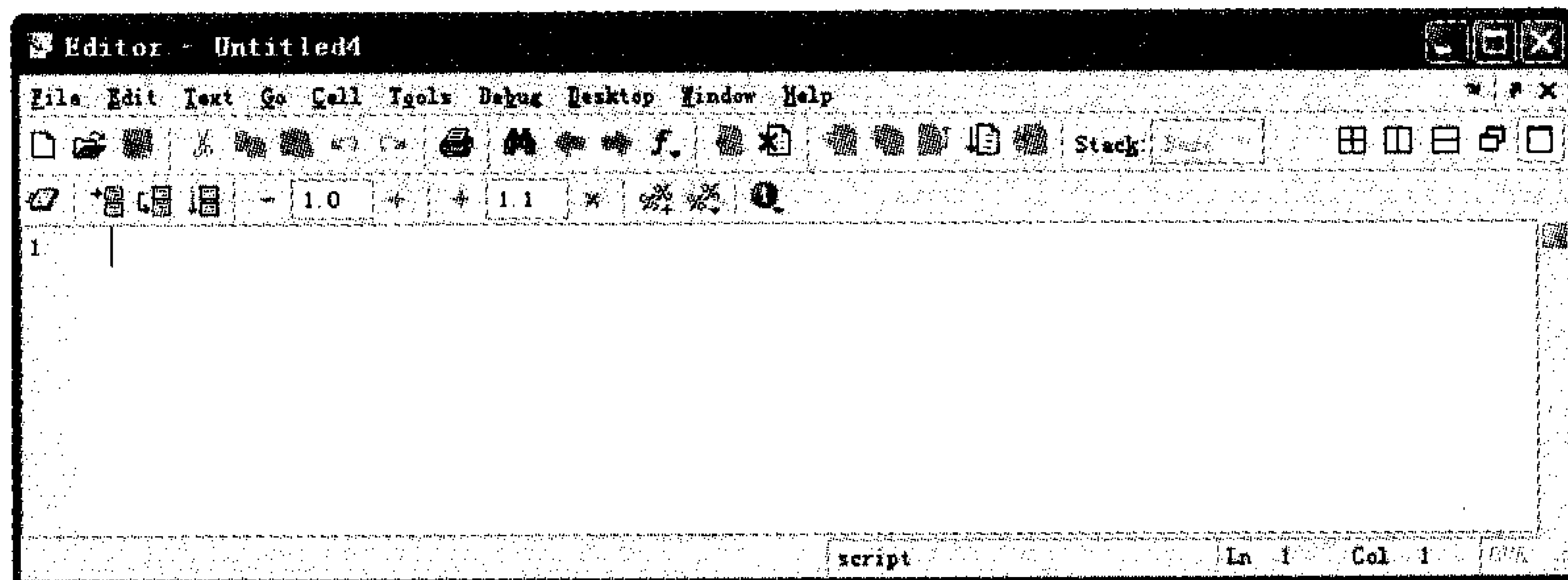


图 1-3 “M-edit 编辑调试”窗口

3）MATLAB 数学函数库

MATLAB 数学函数库是大量的各种形式的数学函数和算法的集合，它不仅包括了最基本的初等函数，如 sum、sine、cosine 和复数运算等，而且包含了大量复杂的高级函数和算法，如贝塞尔（Bassel）函数，快速傅里叶变换和矩阵求逆等。用户在编写自己的 MATLAB 程序时，可以轻松地调用这些函数和算法，从而极大地方便了算法的开发。所有这些函数按类别分别存放在 MATLAB 工具箱目录下的 8 个子目录中，如表 1-2 所示。

表 1-2 MATLAB 数学函数库的分类和组织

目录名	函数功能
elmat	对矩阵和矩阵元素的操作
elfun	初等数学函数
specfun	专门数学函数
matfun	矩阵函数—数值线性代数
datafun	数值分析和傅里叶变换
polyfun	插值和多边形近似
funfun	功能函数和 ODE 求解
sparfun	稀疏矩阵函数

4）图形句柄系统

Handle Graphics®为 MathWorks 公司的注册商标，是 MATLAB 的图形系统。它在包含了大量高级的 2D 和 3D 数据可视化、图形显示、动画生成和图像处理命令的同时，还提供了许多低级的图形命令，允许用户按照自己的需求显示图形和定制应用程序图形用户接口，既方便又灵活。具体的函数分为五大类，分别放置在 MATLAB 工具箱下 5 个不同的目录内，如表 1-3 所示。

表 1-3 MATLAB 图形函数的分类和组织

目录名	函数功能
graph2d	二维图形函数
graph3d	三维图形函数
specgraph	专门图形函数
graphics	图形句柄函数
uitools	图形用户界面工具

5）MATLAB 应用程序接口

MATLAB 的外部接口使得 MATLAB 可与外部设备和程序实现数据交互和程序移植，可以扩充 MATLAB 强大的数值计算和图形显示功能，从而弥补了其执行效率较低的缺点，同时增强了其他应用程序进行软件开发的功能，提高了软件开发效率。MATLAB 接口工具不仅使得 MATLAB 可以十分方便地与其他应用程序交换数据和信息，还实现了与其他程序函数和算法的交互。所以，通过 MATLAB 接口编程，可以充分利用现有资源，能更容易地编写出功能强大、结构简洁的应用程序。MATLAB 主要提供了 MEX 文件、MAT 文件、MATLAB 计算引擎、COM 和 DDE、Web 服务、硬件接口和 Excel 生成器、Java 生成器和.NET

生成器等形式的接口。

1.2 MATLAB 软件安装、界面和帮助

对于 MATLAB 初学者来说，掌握 MATLAB 软件如何安装、熟悉 MATLAB 编程环境，以及了解如何使用帮助文档至关重要。本节将介绍这三方面的内容，使读者能够尽快地掌握 MATLAB，并且能够利用该软件在工程中解决实际问题。

1.2.1 MATLAB R2007a 系统软、硬件资源的要求

根据多年的编程经验，这里给出的配置比官方的要高，这样可以保证一定的运行速度。

- CPU：最好是 Pentium IV 2GHz 以上。
- 内存：最低要求是 512 MB，最好是 1 GB 或以上。
- 硬盘：至少预留 5 GB 以上的专用空间。
- 显卡：最小为 16 位 256 色或以上的图形适配器，最好是 24 位或 32 位 OpenGL 图形适配器，要安装 DirectX 9.0c 或更新版本。
- 系统：Windows 2000/2003/XP 或其他相关产品。
- 光驱：20 倍速以上光驱或安装 Daemon Tools V3.47 或以上版本虚拟光驱。
- 预装软件如下：
  - 安装 office 2000/2002/2003/2007，用以运行 MATLAB 的 Notebook、Excel Builder 和 Excel Link 等软件。
  - Adobe Acrobat Reader 6.0 及以上版本的 PDF 文件浏览器。
  - 其他用于接口的软件。

1.2.2 MATLAB 软件安装

本书主要介绍 MATLAB R2007a 在具有 Windows 2000/XP 操作系统上的 PC 机上的安装过程，在安装 MATLAB 前，需要做以下准备工作。

- 关闭所有正在运行的病毒检测软件，待安装完成后再重新启动病毒检测软件。
- 退出正在运行的其他程序，特别是退出 MATLAB 的其他版本或副本。
- 检查虚拟光驱等计算机软件是否处于良好状态。
- 抄写好 MATLAB 的产品注册码备用。

用安装好的虚拟光驱将 MATLAB R2007a 镜像文件载入，按照提示进行安装即可。安装完成后，会发送一个 MATLAB R2007a 启动快捷方式到桌面，如图 1-4 所示。通过此快捷方式，用户就可以运行 MATLAB R2007a 了。



图 1-4 MATLAB R2007a 启动快捷方式

1.2.3 认识 MATLAB R2007a 环境

在桌面上双击图 1-4 所示的快捷方式，进入 MATLAB R2007a 工作界面，如图 1-5 所示。MATLAB R2007a 的工作界面包括 7 个窗口，它们是主窗口、命令窗口、命令历史记录窗口、当前目录窗口、工作空间窗口、帮助窗口和评述器窗口。下面简要说明各主要窗口的功能。

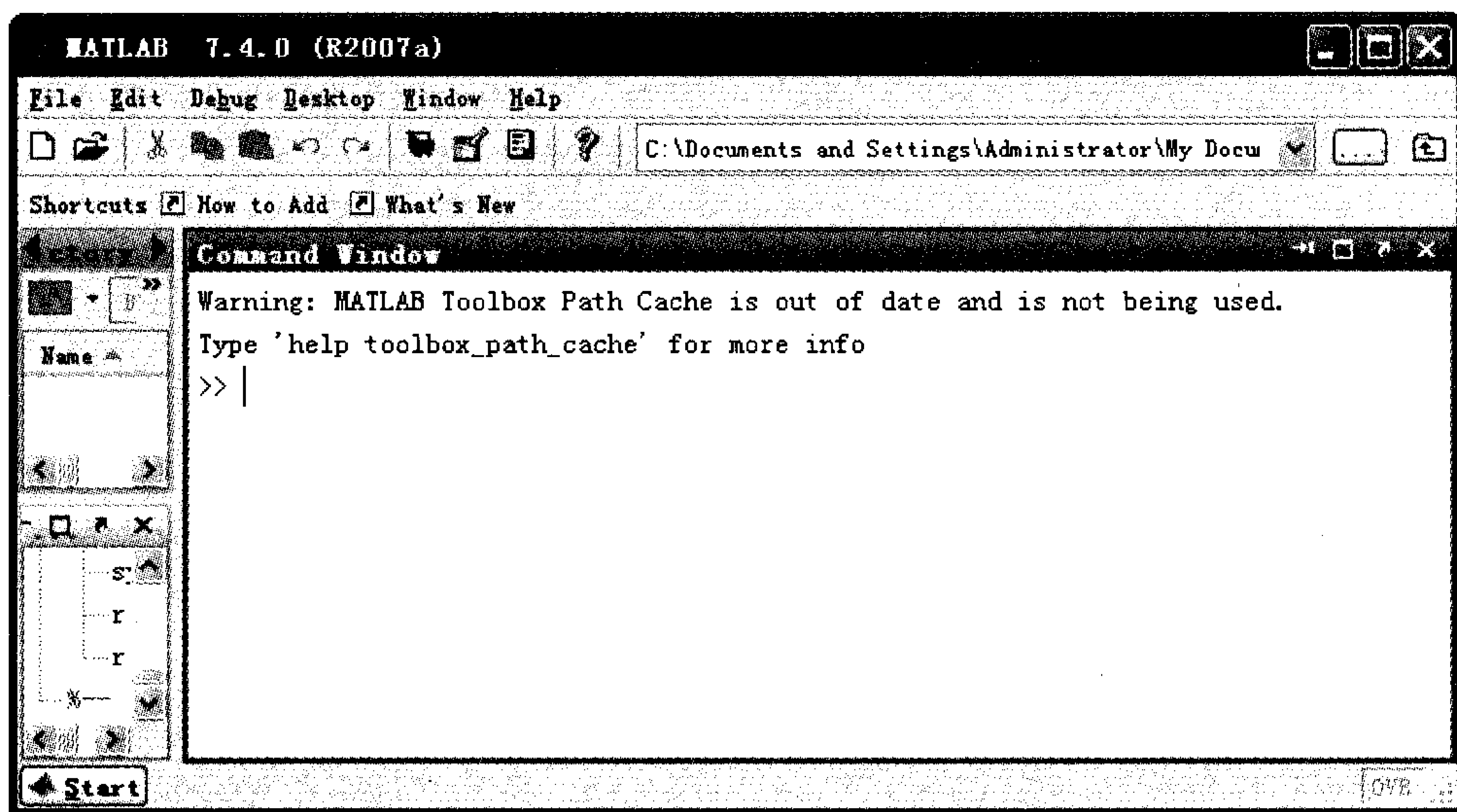


图 1-5 MATLAB R2007a 工作界面

1. 主窗口

MATLAB R2007a 的主窗口兼容其他 6 个子窗口，还包含 6 个菜单和 1 个工具栏。

1）菜单栏

（1）单击菜单栏中的【File】菜单，将弹出如图 1-6 所示的菜单项，其中各项功能如下。

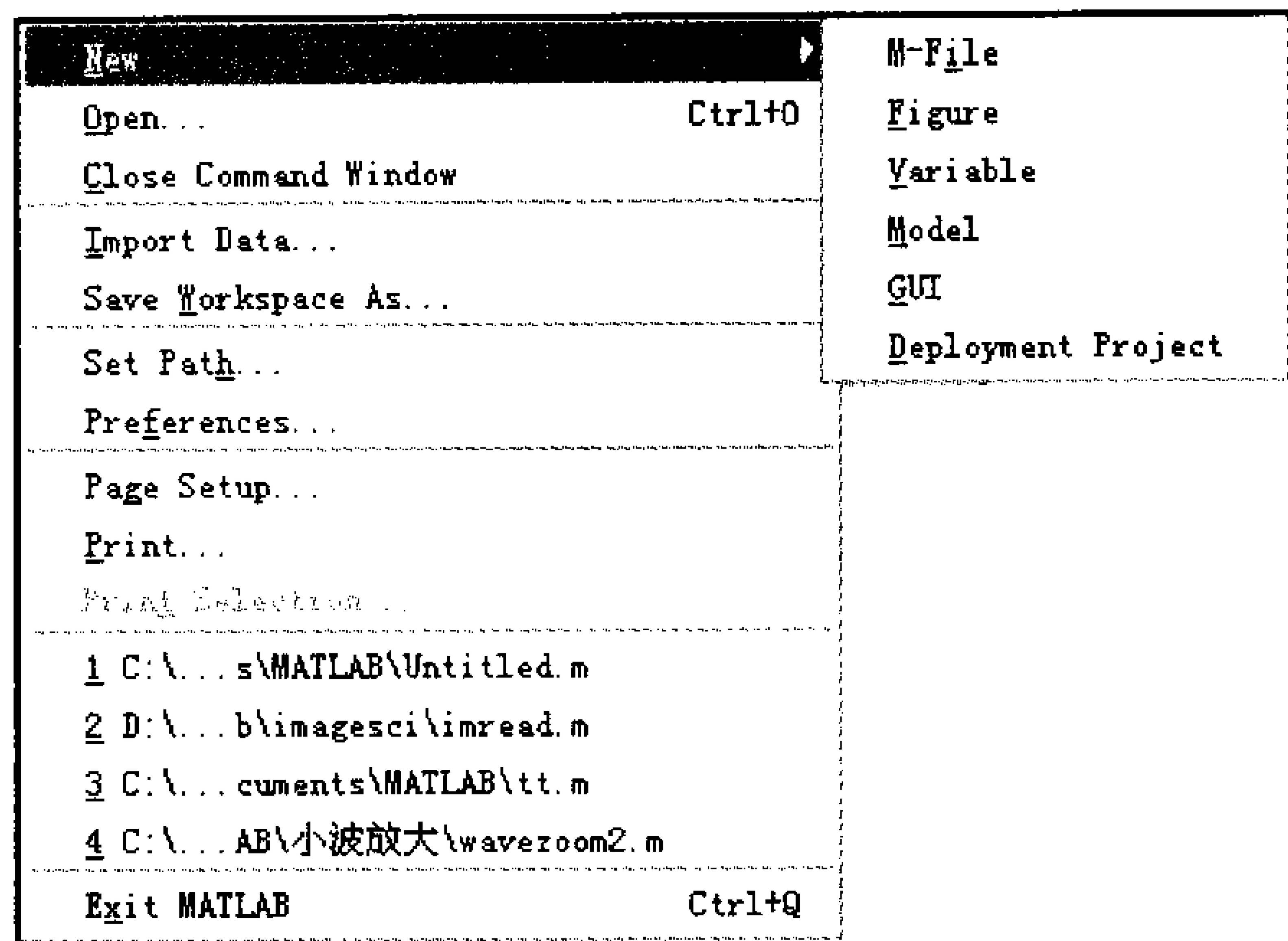


图 1-6 【File】菜单项

- “New”选项包括“M-File”、“Figure”、“Variable”、“Model”和“GUI”5个选项，单击“M-File”选项打开M文件编辑器；单击“Figure”选项打开一个空白的图形窗口；单击“Variable”选项打开工作空间窗口；单击“Model”选项打开创建新模型的窗口；单击“GUI”选项打开创建新的图形用户界面对话框。
- “Open...”选项用来打开“Open”对话框，用户可在对话框中选择想要打开的文件，MATLAB将用相应的编辑器来选定的文件打开。
- “Close Command Window”选项用来关闭当前的活动窗口。
- “Import Data...”选项用来打开“Import”对话框，在该对话框中用户可以选择相应的数据文件，将该文件中的数据导入到MATLAB的工作空间。
- “Save Workspace As...”选项用来打开一个存储MAT文件的对话框，用户将要保存的工作空间命名以后进行存储。
- “Set Path...”选项用来打开页面对话框，在该对话框中，用户可以设置页面的布局、页面的页眉和页面中所用的字体等。
- “Print...”选项用来设置打印参数，打印选定的页面内容等。
- “Print Selection...”选项用来打印命令窗口内选定的内容。打印方法是先选定命令窗口中的要打印的内容，然后单击该选项，打开“打印”对话框，确定打印参数后打印。
- “Exit MATLAB”选项用来关闭MATLAB，按【Ctrl+Q】组合键也可以关闭MATLAB。

（2）单击菜单栏中的【Edit】菜单，将弹出如图1-7所示的菜单项，其中各项功能如下。

Undo Ctrl+Z
Redo
Cut Ctrl+W
Copy Alt+W
Paste Ctrl+Y
Paste to Workspace...
Select All
Delete Ctrl+D
Find...
Find Files...
Clear Command Window
Clear Command History
Clear Workspace

图 1-7 “Edit”菜单项

- “Undo”选项用来撤销上一次的操作。
- “Redo”选项用来重复上一次的操作。
- “Cut”选项用来剪切预先选定的内容。
- “Copy”选项用来复制预先选定的内容。
- “Paste”选项用来将预先存放在缓冲区的内容粘贴到光标所在位置。
- “Paste to Workspace...”选项用来打开导入数据向导，引导用户存放在缓冲区的内容按某一特定格式存放到剪贴板变量当中。
- “Select All”选项用来选定当前窗口中的所有内容。
- “Delete”选项用来删除预先选定的内容。
- “Find...”选项用来打开“查找”对话框，可以在当前命令窗口、当前目录或当前目录中 M 文件内查找相应内容。
- “Find Files...”选项用来查找文件。
- “Clear Command Window”选项用来清除命令窗口中显示的全部内容，但不会删除当前工作空间中的变量。
- “Clear Command History”选项用来清除历史记录窗口中显示的全部内容。
- “Clear Workspace”选项用来清除工作空间中的全部内容。

（3）单击菜单栏中的【Debug】菜单，将弹出如图 1-8 所示的菜单项，其中各项功能如下。

✔ Open M-Files when Debugging
Step F10
Step In F11
Step Out Shift+F11
Continue F5
Clear Breakpoints in All Files
Stop if Errors/Warnings...
Exit Debug Mode Shift+F5

图 1-8 “Debug”菜单项

- “Open M-Files when Debugring”选项用来打开一个要调试的 M 文件。
- “Step”选项表示逐个语句进行调试。
- “Step In”选项表示进入下一个语句。
- “Step Out”选项表示退出调试。
- “Continue”选项表示继续调试。
- “Clear Breakpoints in All Files”选项表示清除所有文件的断点。
- “Stop if Errors/Warnings…”选项表示遇到错误和警告断点就停止运行。
- “Exit Debug Mode”选项表示退出调试模式。

（4）单击菜单栏中的【Desktop】菜单，将弹出如图 1-9 所示的菜单项，其中各项功能如下。

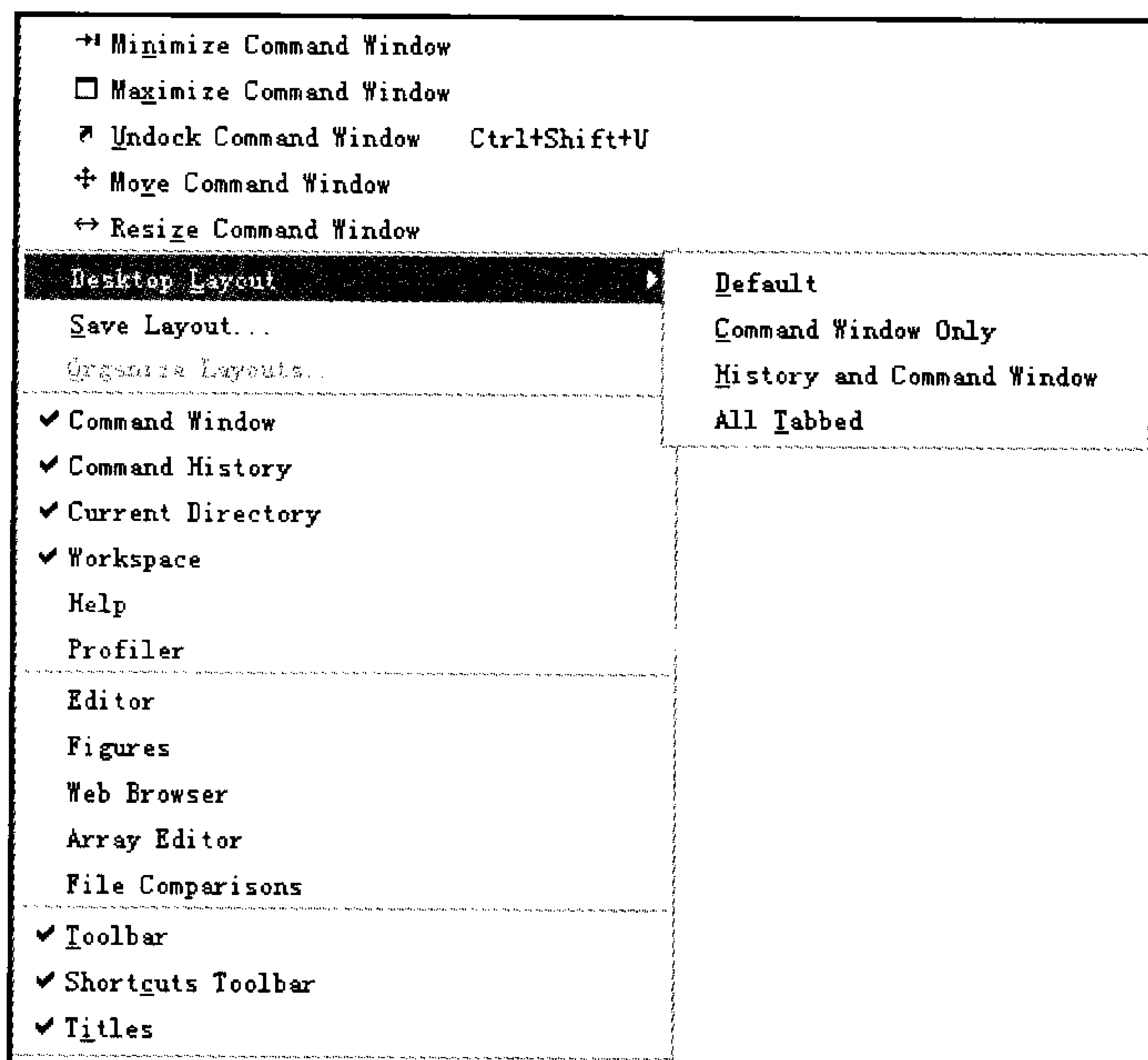


图 1-9 “Desktop”菜单项

- “Minimize Command Window”选项表示用来最小化命令窗口。
- “Maximize Command Window”选项表示用来最大化命令窗口。
- “Undock Command Window”选项表示用来取消对命令窗口的操作。
- “Move Command Window”选项表示用来移除命令窗口。
- “Resize Command Window”选项表示用来重新调整命令窗口的大小。
- “Desktop Layout”选项表示用来选择窗口布局，它含有 4 个菜单项，分别是“Default”（默认布局）、“Command Windows Only”（只显示命令窗口）、“History and Command Window”（只显示命令窗口和命令历史记录窗口）和“All Tabbed”（同时显示 5 个窗口）。
- “Save Layout…”选项表示存储版式。
- “Organize Layouts…”选项表示组织版式，由用户自定义版面决定哪些窗口显示，哪些窗口不显示。

- 6 个可选的窗口复选菜单项，分别是选项“Command Window”（命令窗口）、“Command History”（命令历史窗口）、“Current Directory”（当前记录窗口或称为路径浏览窗口）、“Workspace”（工作空间窗口）、“Help”（帮助窗口）、“Profiler”（评述器窗口），被选中的菜单项在菜单的左边出现一个对号，被选中的菜单项对应的工具栏或标题将显示出来。
- 5 个复选的编辑器菜单、图形、浏览器菜单，分别是“Editor”（编辑器）和“Figures”（多图形窗口）、“Web Browser”（网页浏览器）、“Array Editor”（数组编辑器）和“File Comparisons”（文件比较器），被选中的菜单项在菜单的左边出现一个对号，被选中的菜单项对应的编辑器、多图形窗口、浏览器将显示出来。
- 3 个复选的功能复选菜单，分别是“Toolbar”（主窗口工具栏）、“Shortcuts Toolbar”（子窗口工具栏）和“Titles”（子窗口标题），同样，被选中的菜单项在菜单的左边出现一个对号，被选中的菜单项对应的工具栏或标题将显示出来。

（5）单击菜单栏中的【Windows】菜单，将弹出如图 1-10 所示的菜单项，其中各项功能如下。

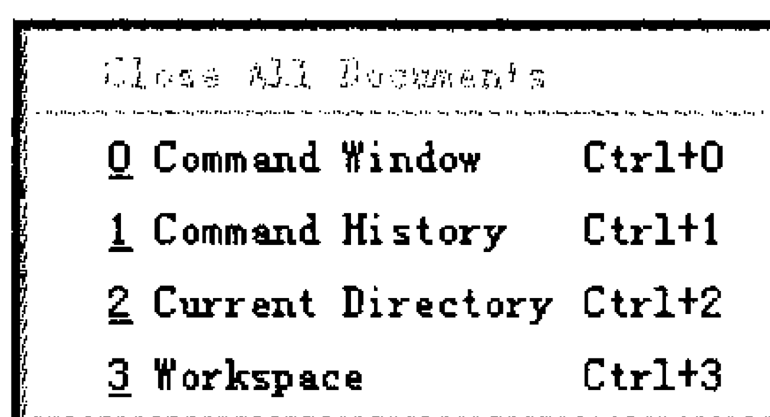


图 1-10 “Windows”菜单项

- “Close All Documents”选项用来关闭所有打开的编辑器窗口，包括 M-File、Figure、Model 和 GUI 窗口。
- 当某一窗口被打开的时候，MATLAB 将自动在 Windows 菜单栏产生这个窗口的名称，并在名称的左边加一个序号（有下画线），在名称的右边加上将这个窗口变为当前窗口的快捷键。

（6）单击菜单栏中的【Help】菜单，将弹出如图 1-11 所示的菜单，其中各项功能如下。

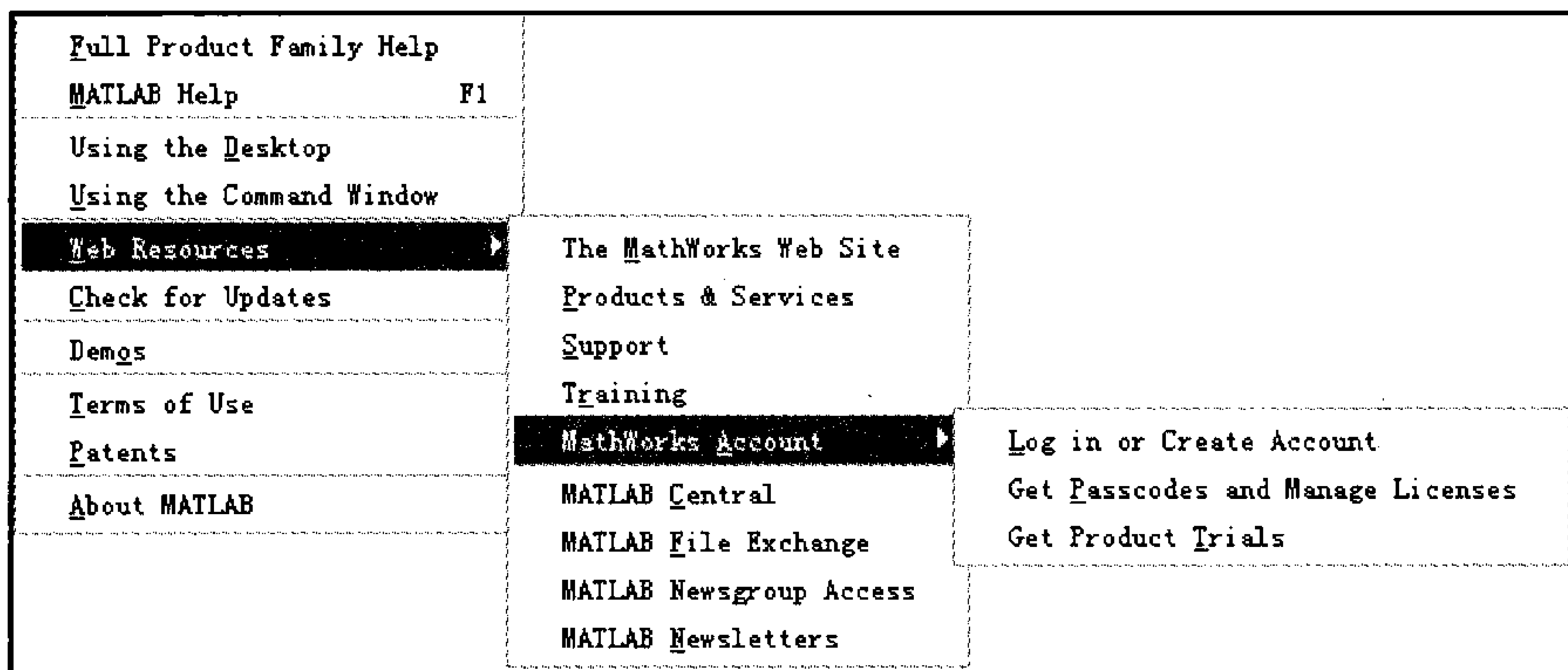


图 1-11 “Help”菜单项

- “Full Product Family Help”选项用来打开所有 MATLAB 系列产品家族的帮助文件。

- “MATLAB Help”选项用来打开 MATLAB 的帮助文件。
- “Using the Desktop”选项用来打开 MATLAB 的帮助文件，并从 Using the Dosktop 开始显示帮助文件。
- “Using the Command Window”选项用来打开 MATLAB 帮助文件，并从 Command Window 开始显示帮助文件。
- “Web Resources”选项共有 8 个子项：单击“The MathWorks Web Site”可以自动链接到 MathWorks 公司的站点；“Products&Services”选项用来网上查阅 MATLAB 的产品发布和服务情况；“Support”选项用来查阅 MATLAB 网上技术支持；“Training”用来查阅 MATLAB 练习；“MathWorks Account”选项表示 MathWorks 说明，共包括“Log in or Create Account”（在服务器上查阅或建立一个说明）、“Get Passcodes and Manage Licenses”（获得产品的注册码和管理许可证）和“Get Product Trials”（获得产品审查）；单击“MATLAB Central”、“MATLAB File Exchange”、“MATLAB Newsgroup Access”和“MATLAB Newsletters”选项中的任一项将链接到相应的网页上。
- “Check for Updates”选项用来网上更新检查和网上更新。
- “Demos”选项用来打开功能演示。
- “Terms of Use”选项用来查看 MATLAB 产品的使用条件。
- “Patents”选项用来查看 MATLAB 产品专利情况。
- “About MATLAB”选项用来打开 MATLAB 版权界面，版权界面有用户安装的 MATLAB 的详细版本号、发行日期、用户名和用户单位名称等。

2）工具栏

MATLAB 主窗口的工具栏（如图 1-12 所示）含有 10 个按钮，从左到右按钮的功能如下。

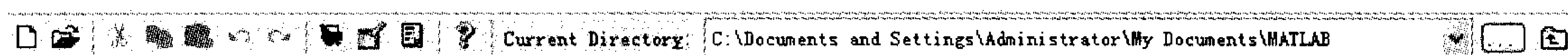


图 1-12　MATLAB 主窗口工具栏

- 新建或打开一个 MATLAB 文件。
- 剪切、复制或粘贴所选定的对象。
- 撤销或恢复上一次的操作。
- 打开 Simulink 主窗口。
- 打开 GUI 窗口。
- 打开 Profiler 窗口。
- 打开 MATLAB 帮助窗口。
- 设置当前路径。

3）窗口开始菜单

在 MATLAB 主窗口的左下角有一个【开始】菜单Start，单击【开始】菜单可以看到里面有 MATLAB、Toolboxes、Simulink、Blocksets、Shortcuts、Desktop Tools 和 Web7 个程序组，还有 Preferences...、Find Files...、Help 和 Demos4 个选项（如图 1-13 所示）。MATLAB 是主程序组（如图 1-14 所示），包含 Excel Link（Excel 链接器）、MATLAB Builder

for .NET（.NET 编译器）、MATLAB Builder for Excel（Excel 编译器）、MATLAB Builder for Java（Java 编译器）、MATLAB Compiler（MATLAB 编译器）、MATLAB Report Generator（MATLAB 报告生成器）、SimBiology 和 SystemTest 8 个程序组，还有 Import Wizard、Profiler、GUIDE（GUI Builder）、Notebook、Plot Tools、Time Series Tools、Help、Demos、MATLAB Central（Web）、Product Page（Web）10 个可选项。



图 1-13 【开始】菜单

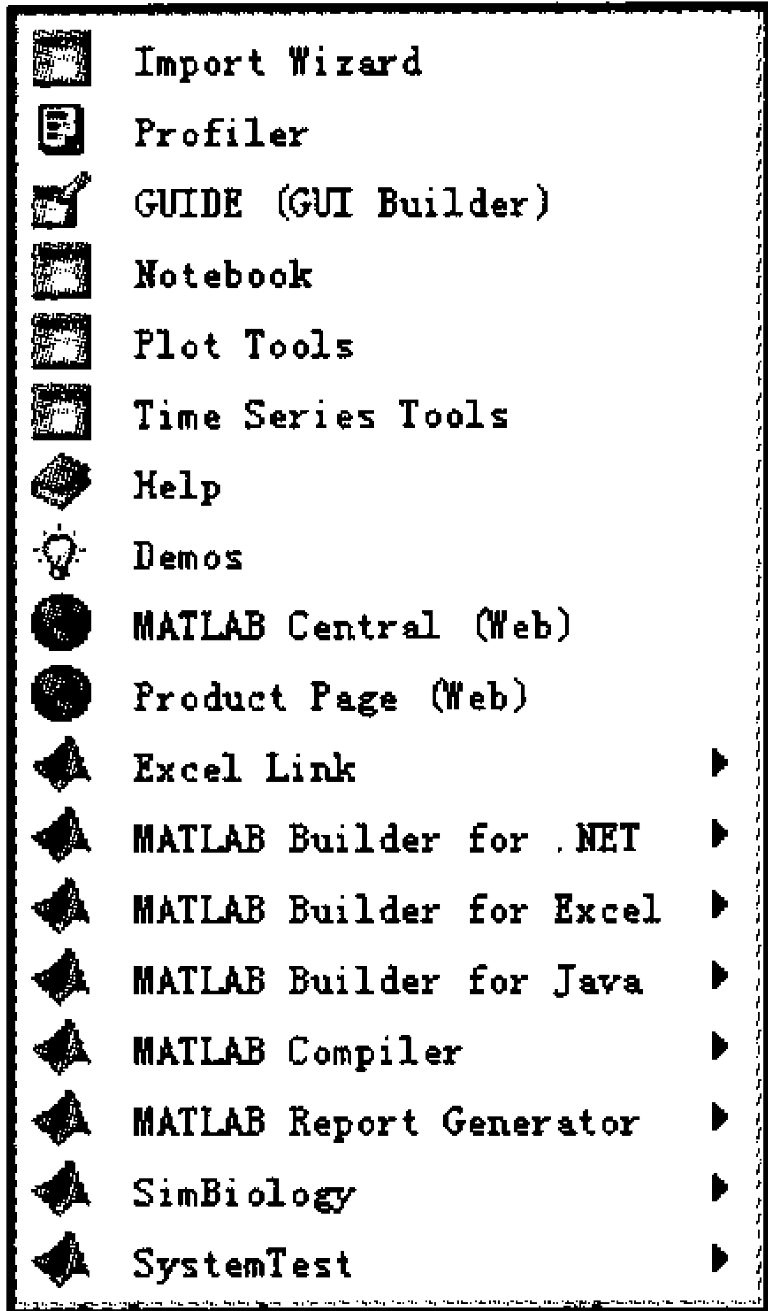


图 1-14 MATLAB 主程序组选项

Toolboxs 是工具箱，包含了所有用户已经安装的工具箱；Simulink 是仿真模块集，含有所有的仿真程序模块；Blocksets 是模块集，含有用户安装的除仿真模块集以外的其他模块集；Shortcuts 中包含了使用帮助、工具栏以及其他方面的使用技巧。Desktop Tools 是桌面工具，含有 Command History、Current Directory、View Sourse Files...、Editor、Path 和 workspace 6 种工具；Web 是一个网页集合，罗列了与 MATLAB 相关的所有网页，以有利于用户搭配，需要时只要单击该选项就可以打开所需要的网页；Preferences...、Find Files、Help 和 Demos 4 个选项功能分别是“优化”、“搜索文件”、“帮助”和“演示”。

2. **命令窗口**

命令窗口（Command Window）如图 1-15 所示。

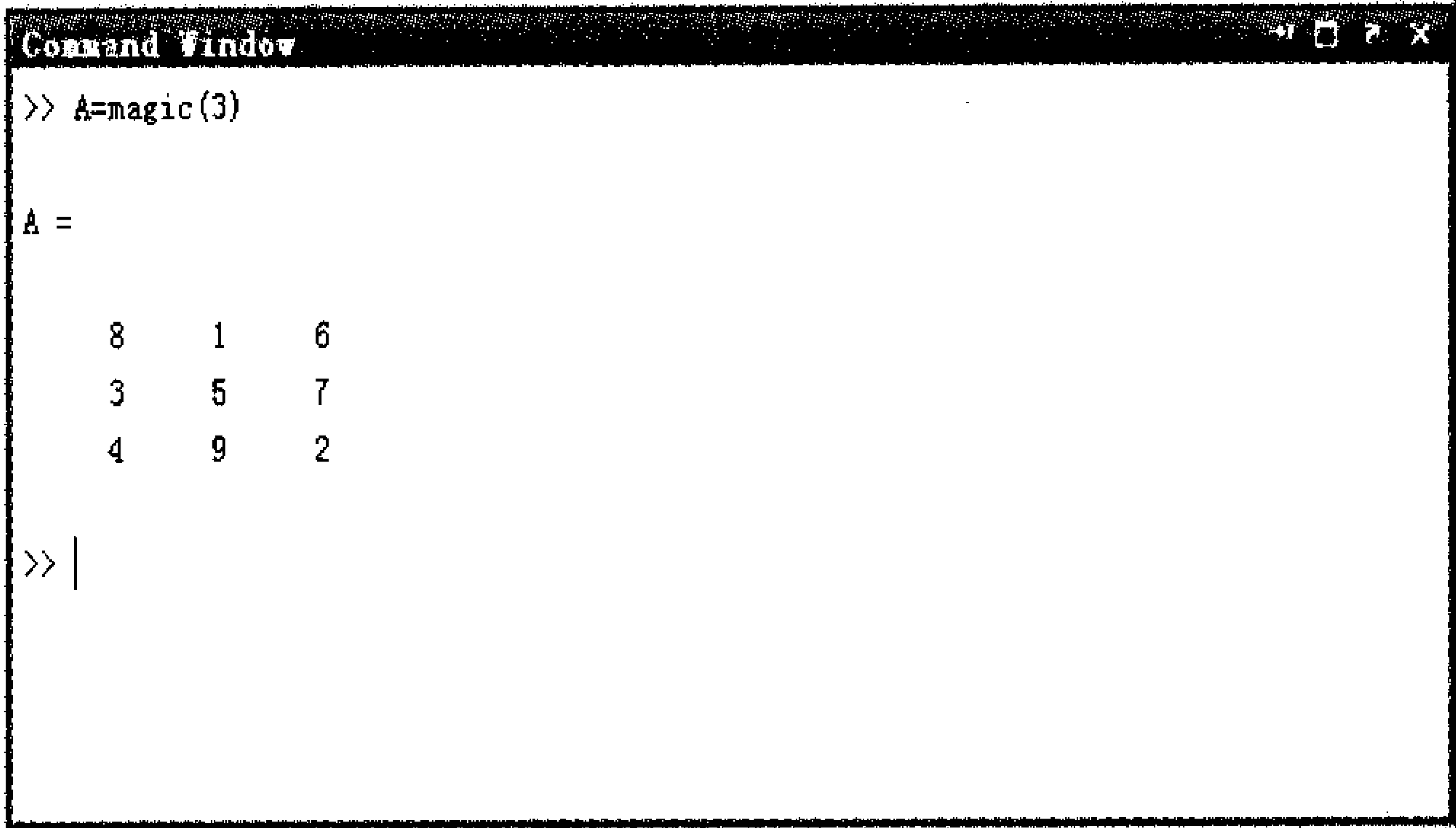


图 1-15 MATLAB 命令窗口

当 MATLAB 启动完成，命令窗口显示以后，窗口处于准备编辑状态。符号“>>”为运算提示符，说明系统处于准备状态。当用户在提示符后输入表达式并按回车键之后，系统将会给出运算结果，然后继续处于系统准备状态。

3. 命令历史记录窗口

命令历史记录窗口（Command History）如图 1-16 所示。在默认情况下，命令历史记录窗口会保留自安装以来所有用过的命令的历史记录，并详细记录了命令使用的日期和时间，为用户提供了所使用过的命令的详细查询，所有保留的命令都可以单击后执行。当执行【Edit】→【Clear Command History】命令后，所有命令记录将被清除。

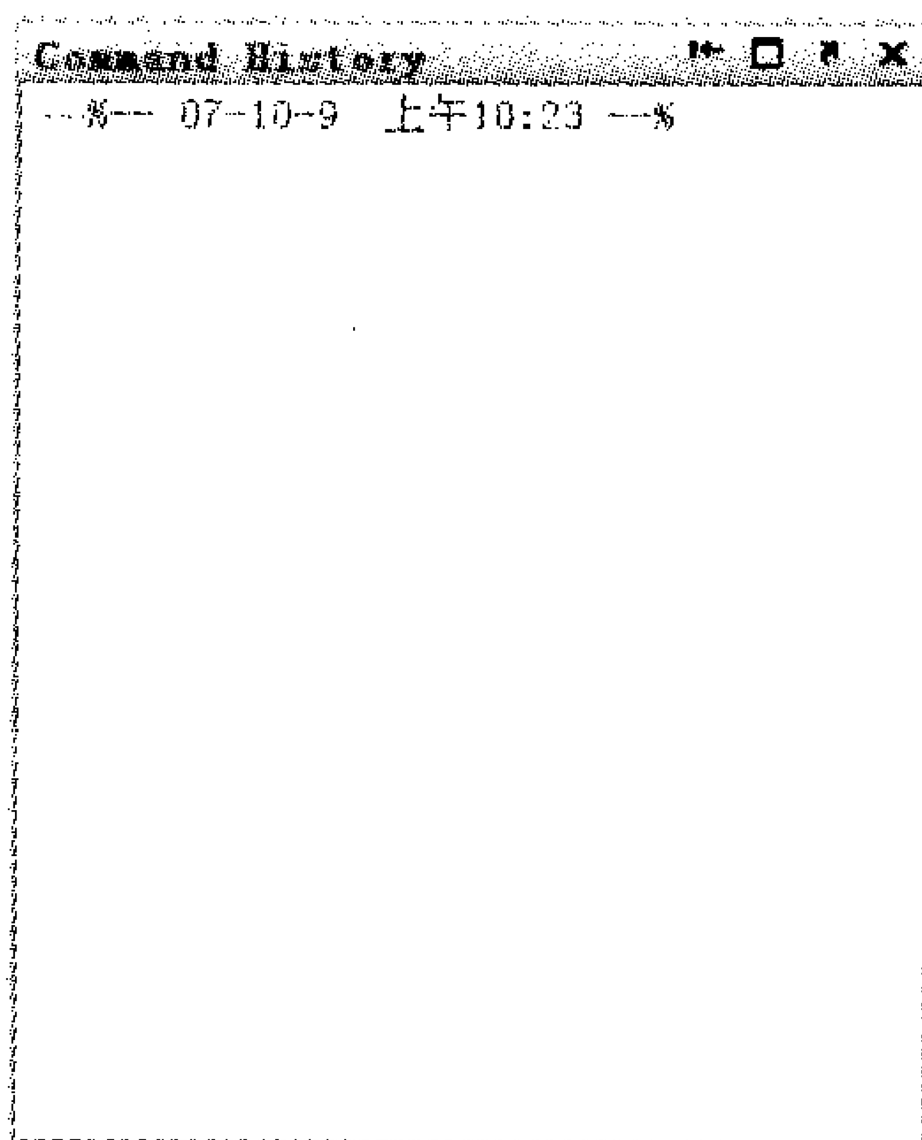


图 1-16　命令历史记录窗口

4. 当前目录窗口

当前目录窗口（Current Directory）如图 1-17 所示，它的主要功能是显示或改变当前目录，不仅可以显示当前目录下的文件，而且还可以提供搜索。通过上面的目录选择下拉菜单，用户可以轻松地选择已经访问过的目录。单击右侧的按钮，可以打开路径选择对话框，在这里用户可以设置和添加路径。也可以通过上面一行超级链接来改变路径。

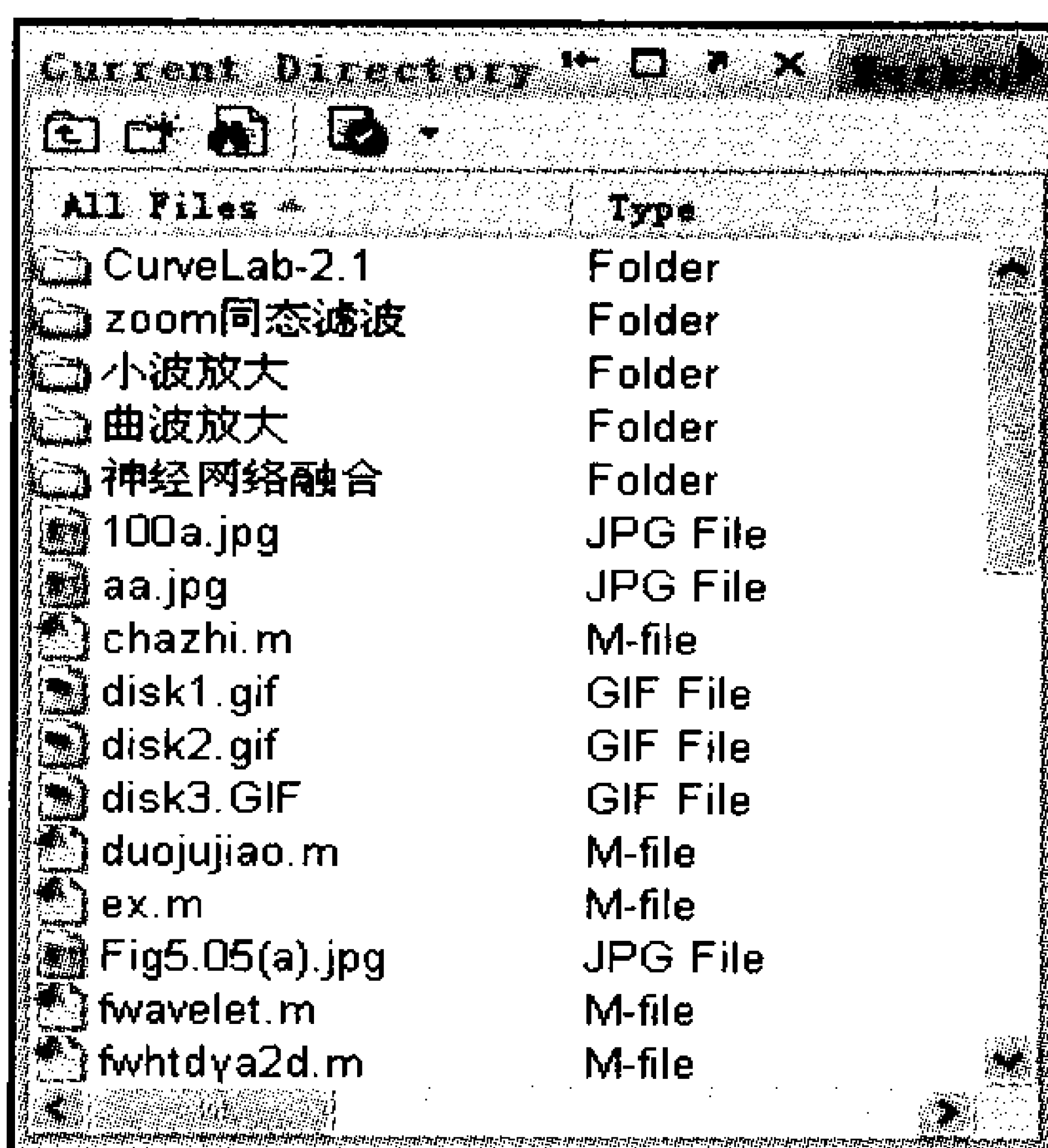


图 1-17　命令历史记录窗口

5. **工作空间窗口**

工作空间窗口（Workspace）如图1-18所示，它是MATLAB的一个重要组成部分。该窗口有显示目前内存中存放的变量名、变量存储数据的维数、变量存储的字节数和变量类型说明等的功能。工作空间窗口有自己的工具栏，从左到右按钮的功能依次是新建变量、打开选择的变量、载入数据文件、保存、打印和删除等。

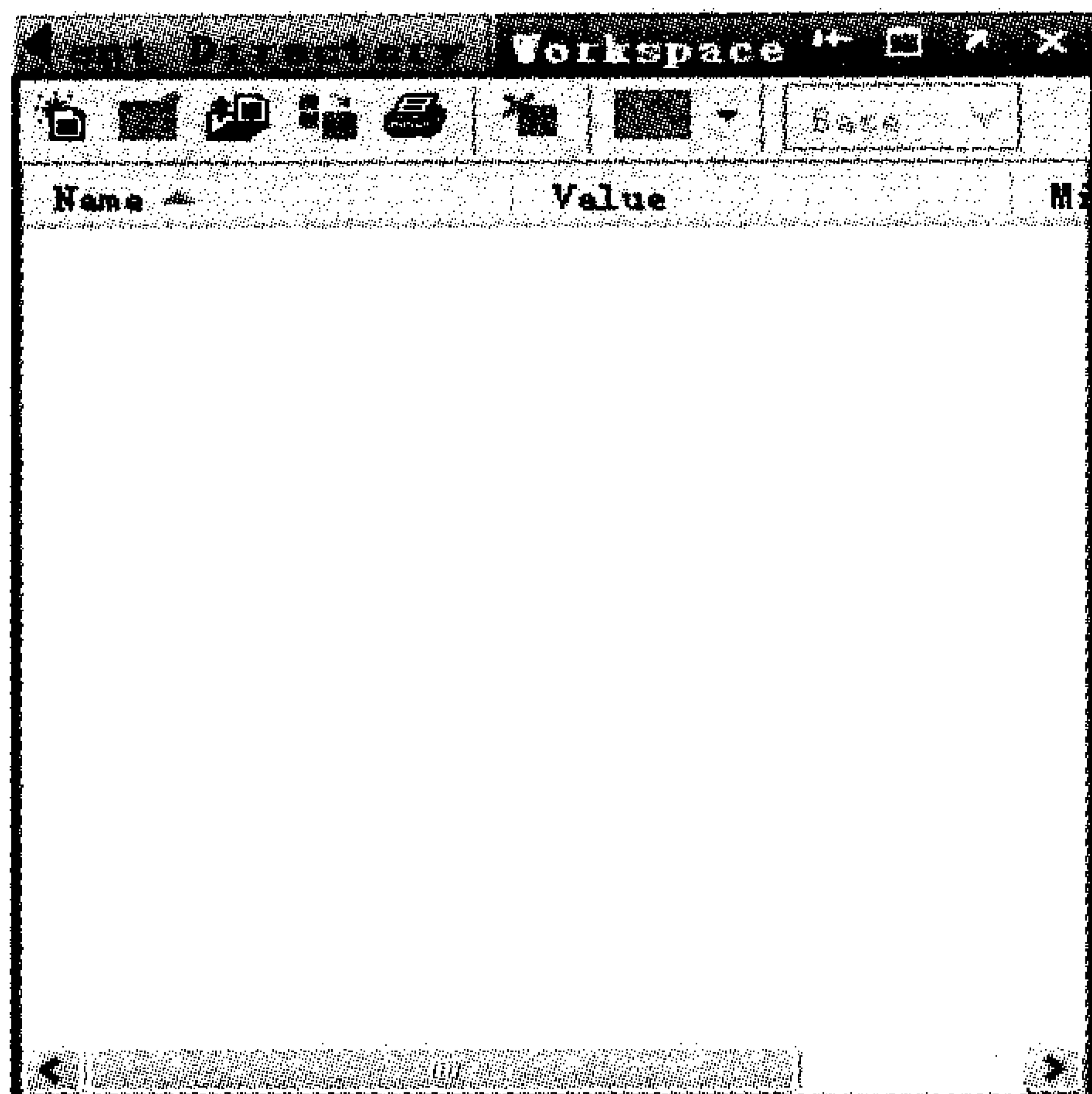


图1-18 工作空间窗口

1.2.4 MATLAB R2007a的帮助系统

MATLAB R2007a的帮助系统非常强大，是该软件的信息查询、联机帮助中心。MATLAB的帮助系统主要包括联机帮助系统、联机演示系统、远程帮助系统和命令查询系统。

1. **联机帮助系统**

MATLAB R2007a的联机帮助系统可通过以下4种方式打开。

- 打开主窗口以后按【F1】键。
- 在主窗口中单击工具栏上的【问号】按钮。
- 通过选择“帮助”下拉菜单的帮助选项。
- 在命令窗口中输入命令helpdesk、helpwin和doc命令。

打开联机帮助系统以后，系统会出现如图1-19所示的联机帮助窗口。联机帮助窗口包含帮助向导页面和显示页面两部分。帮助向导页面中含有4个可供用户选择的表单窗口，分别是Contens（帮助目录）、Index（帮助索引）、Search Results（查询结果）和Demos（演示系统）。在帮助目录窗口中系统列出了帮助系统中所有大大小小的目录，用户可轻易地找到自己所需要查询的标题，单击该标题就会在帮助显示页面显示该标题的内容，并可以随时联机更新。帮助索引窗口由英文字母索引、标题词索引输入框和索引显示框组成，用户可单击检索内容标题的第一个字母或在标题词索引文本框中输入标题词进行检索。单击相

应字母或输入标题词后在索引显示框会显示检索的内容标题，显示方式是左边显示标题词名称，右边显示该内容所属的产品模块。单击该标题就会在帮助显示页面显示帮助的相应内容。在查询结果窗口，用户可直接在检索词文本框“Search for:”中输入检索词，单击【go】按钮后，帮助系统就会在帮助显示页面显示检索到的内容。帮助显示页面由 1 个有 5 个按钮的工具栏、1 个标题框和 1 个帮助文本显示框组成。工具栏中从左到右 5 个按钮的功能依次是“向前”、“向后”、“刷新”、“打印”和“查找”。

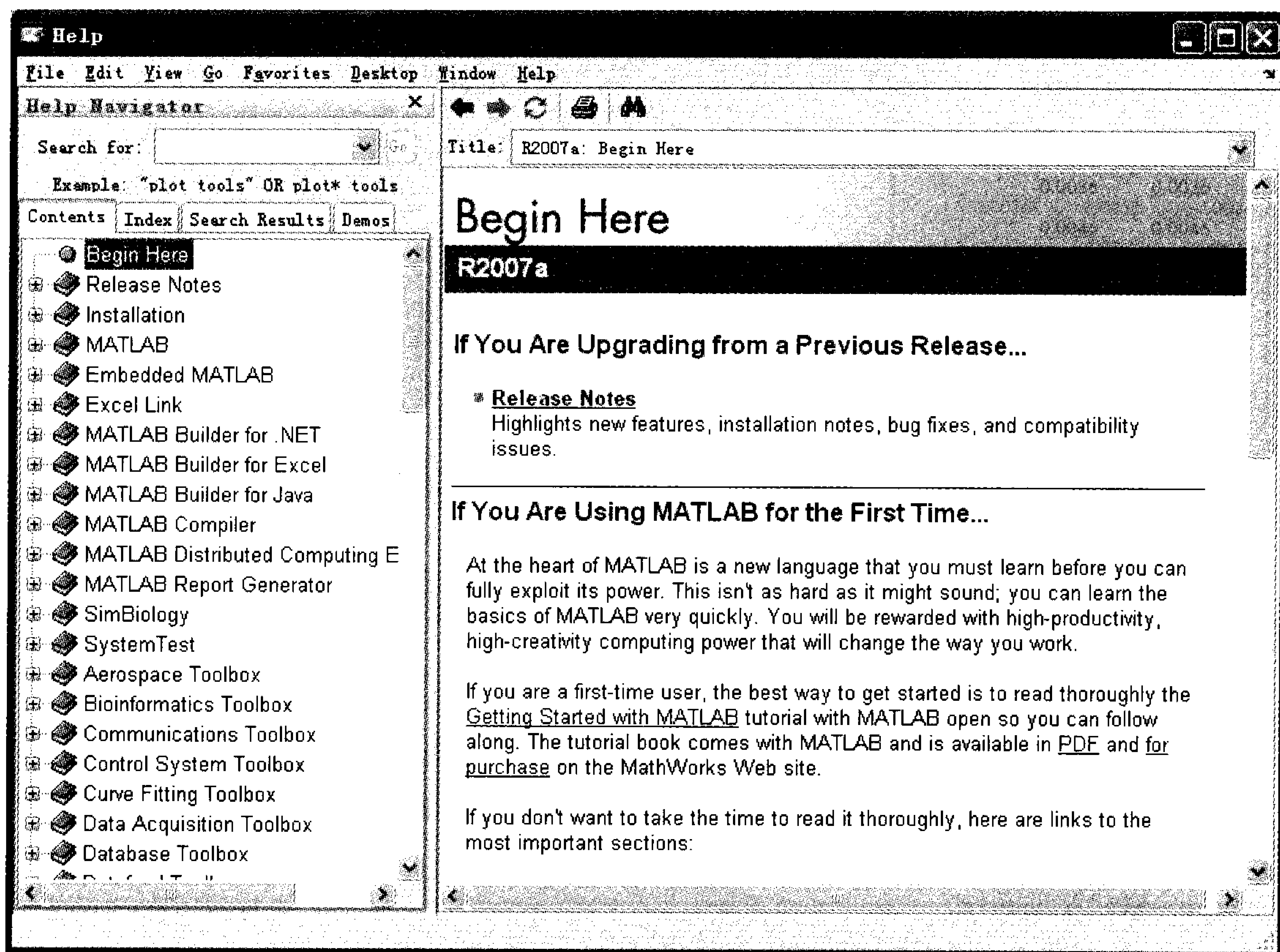


图 1-19　联机帮助窗口

2．联机演示系统

MATLAB R2007a 的联机演示系统可通过以下 3 种方式打开。

- 在主窗口的【Help】菜单下选择“Demos”选项。
- 在帮助目录窗口选择“Demos”表单。
- 在命令窗口输入“demo”命令。

打开后的联机演示窗口如图 1-20 所示。在联机演示窗口的左边是帮助向导页面 Demos 表单窗口，共有 1 个开始说明和 4 个目录，4 个目录分别包含了主程序内容（MATLAB）演示模块、工具箱（Toolbox）演示模块、仿真（Simulink）演示模块和模块集（Blocksets）演示模块。当用户选定所要演示的内容后，只要单击帮助显示窗口里相关标题的超级链接

或单击左侧相关标题即可演示相关内容。

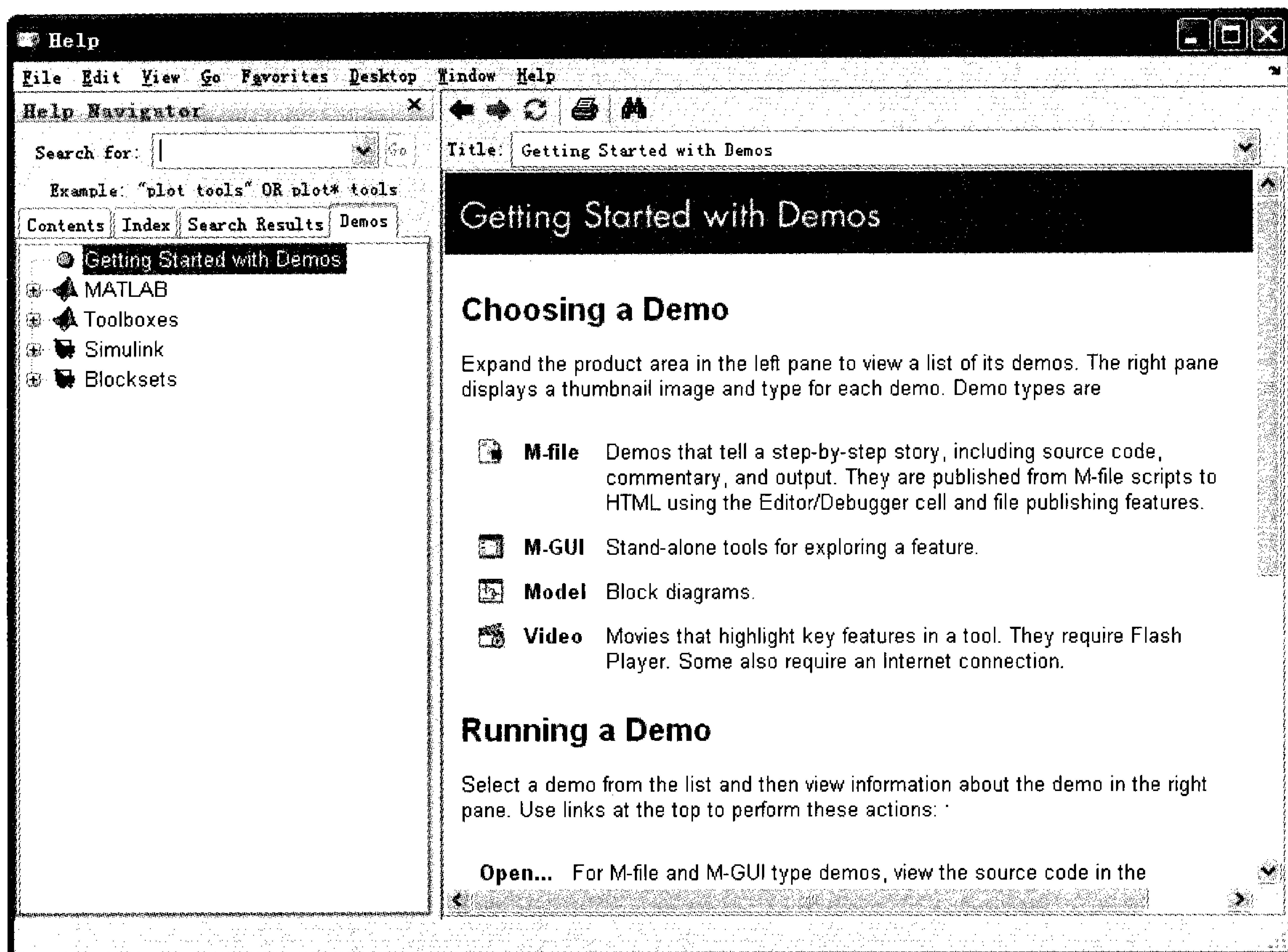


图 1-20 联机演示窗口

3. 远程帮助系统

MATLAB R2007a 的远程帮助系统由网络资源（Web Resources）和更新检查（Check for Updates）两部分组成，可通过选择主窗口中【Help】菜单中这两个选项打开。“Web Resources”中只要用户选择某一项链接就可以直接链接到相应的网站或网页。更新检查选项用来进行 MATLAB 各组件的更新检查和更新。当用户的计算机联网并选择了该项以后，稍后（与计算机配置和网络速度有关）帮助系统会弹出如图 1-21 所示的对话框，即系统经过远程查询以后告诉用户目前各组件的安装版本是不是最新版本，如果是，则显示“Up to date”；如果不是，则显示当前的最新版本号。如果没有找到该组件，则显示“Not found”。该页面的右下方有两个按钮，单击【Check for Updates】按钮用来登录 MathWorks 公司网站下载最新版本组件，单击【Close】按钮则关闭该页面。

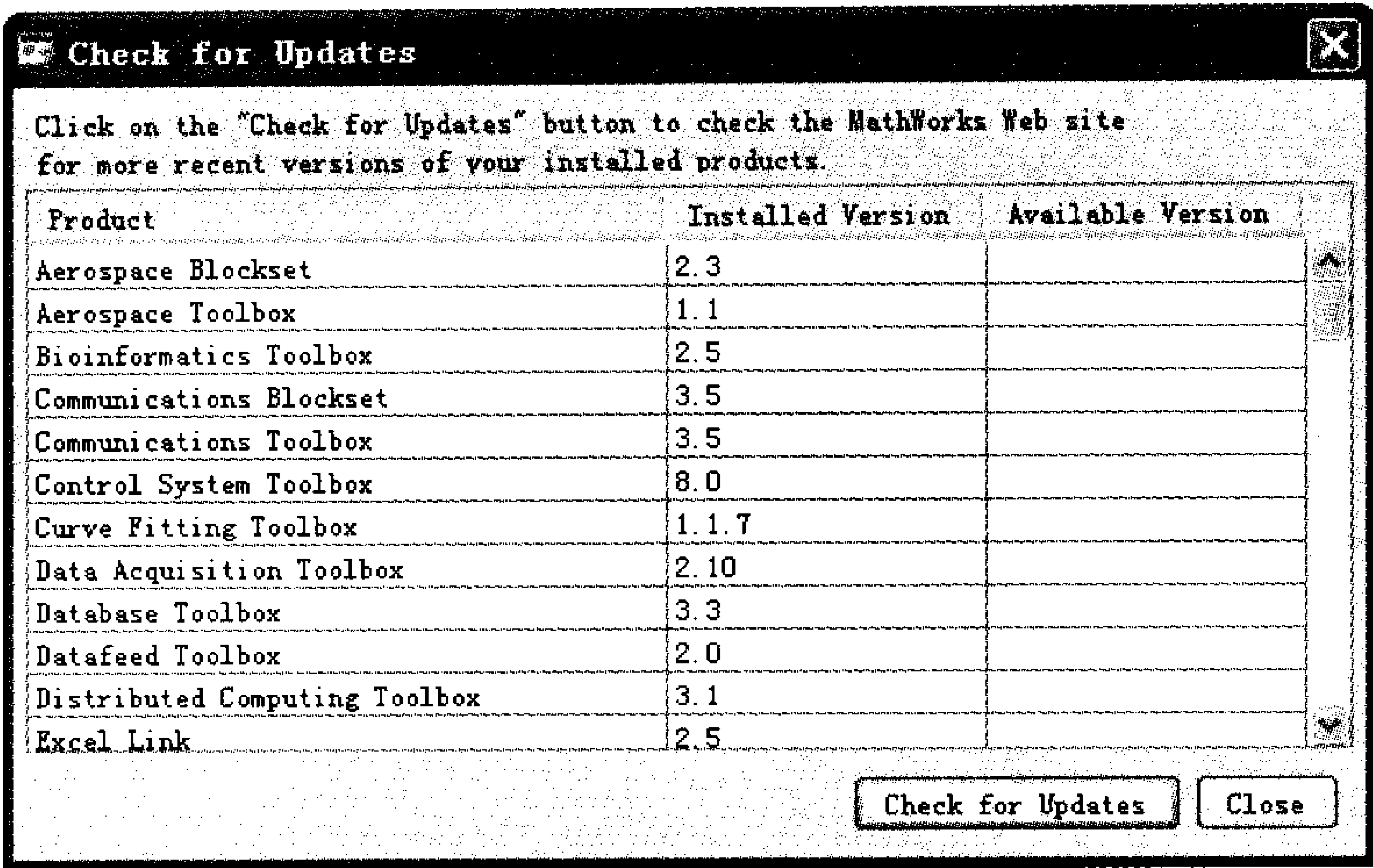


图 1-21　“更新检查”对话框

4．命令查询系统

在实际使用过程中需要启动帮助系统查询的时候，最简单、快捷的办法还是运用 MATLAB 的命令查询系统。MATLAB 有强大的命令查询系统，用户只要在命令窗口输入相应命令，就可以方便地查询所需要的资料。这些命令有 help、help+<函数名>和 lookfor+<关键字>等，详细帮助命令及功能如表 1-4 所示。

表 1-4　帮助命令功能表

命　令	功　能
help	用于显示帮助系统中的所有项目，用目录的形式列出
help+<函数名>	用于查询与该函数有关的帮助内容
lookfor+<关键字>	当函数名称未知时，根据名称或功能关键字查询有关函数
demo	打开演示窗口
info	显示 MATLAB 的一般信息
whatsnew	列出 MATLAB 的新有特征

1.3　通过实例了解 MATLAB

通过前面介绍的 MATLAB 如何安装和查询帮助，对 MATLAB 工作环境有了一定的了解，下面就通过几个实例来加深对 MATLAB 的了解。

1.3.1　命令行程序

在命令窗口，用户可以直接调用 MATLAB 内部已经编译好的 M 文件，也可以直接在命令行提示符下输入命令，然后按回车键运行。下面通过一个矩阵操作的例子来介绍命令行程序。

例程 1.1 为在命令窗口中定义两个矩阵，进行各种矩阵运算。其中函数 magic 用于生成一个魔术矩阵 ***A*** 。

例程 1.1　命令行程序示例

```
>> A=magic(3)    %定义一个魔术矩阵 A
```

得到结果如下：

```
A =
     8     1     6
     3     5     7
     4     9     2
```

输入以下语句生成一个和 ***A*** 一样大小全 1 矩阵 ***B*** 。

```
>> B=ones(3)    %定义全 1 矩阵 B
```

得到结果如下：

```
B =
     1     1     1
     1     1     1
     1     1     1
```

将 ***A*** 、 ***B*** 两个矩阵相加，得到结果如下：

```
>> A+B    %计算符号矩阵加法 A+B
ans =
     9     2     7
     4     6     8
     5    10     3
>>
```

在命令窗口中可以使用 MATLAB 工具箱函数对矩阵进行操作。使用 flipud 函数可以对矩阵进行上下翻转，以前面定义的魔术矩阵 ***A*** 为例。

```
>> flipud(A)
ans =
     4     9     2
     3     5     7
     8     1     6
```

得到结果如下，可以看到，结果与前面分析的一样。而 fliplr 函数可以对矩阵进行左右翻转。

下面针对函数 humps 来介绍 MATLAB 在科学计算中的应用。该函数表达式如下：

$$\text{humps}(x)=\frac{1}{(x-0.3)^2+0.01}+\frac{1}{(x-0.9)^2+0.04}-6$$

下面的代码利用函数 fzero 分别找出了 humps 函数在 x=1.3 附近的零点位置。

例程 1.2　MATLAB 在科学计算中的应用

```
>> format long
>> H_humps=@humps
>> x=fzero(H_humps,1.3)
```

得到结果如下：

```
x =
    1.299549682584822
```

现在要计算它在 $x\in[-1,2]$ 时的面积。在 MATLAB 命令窗口，用户只要输入如下命令即可。

例程 1.3　利用 MATLAB 求面积

```
>> x=linspace(-1,2,100);
>> y=humps(x);
>> format long
>> area=trapz(x,y)
```

其中函数 linspace 将–1 到 2 之间的数值 100 等分（产生间隔均匀的 100 个抽样点），函数 trapz 将根据均匀间隔的抽样值列表，使用梯形分割来近似估计函数的面积（积分）。得到结果如下所示。

```
area =
     26.344731195245956
```

以上只是利用 MATLAB 进行科学计算的一个例子，关于其详细内容将在本书后面的章节进行介绍。

1.3.2　MATLAB 绘图

还是利用上面的 humps 函数，使用 MATLAB 绘图函数可以方便、快捷地做出我们需要的图形。

```
>> plot(x,y)
```

所得结果如图 1-22 所示。

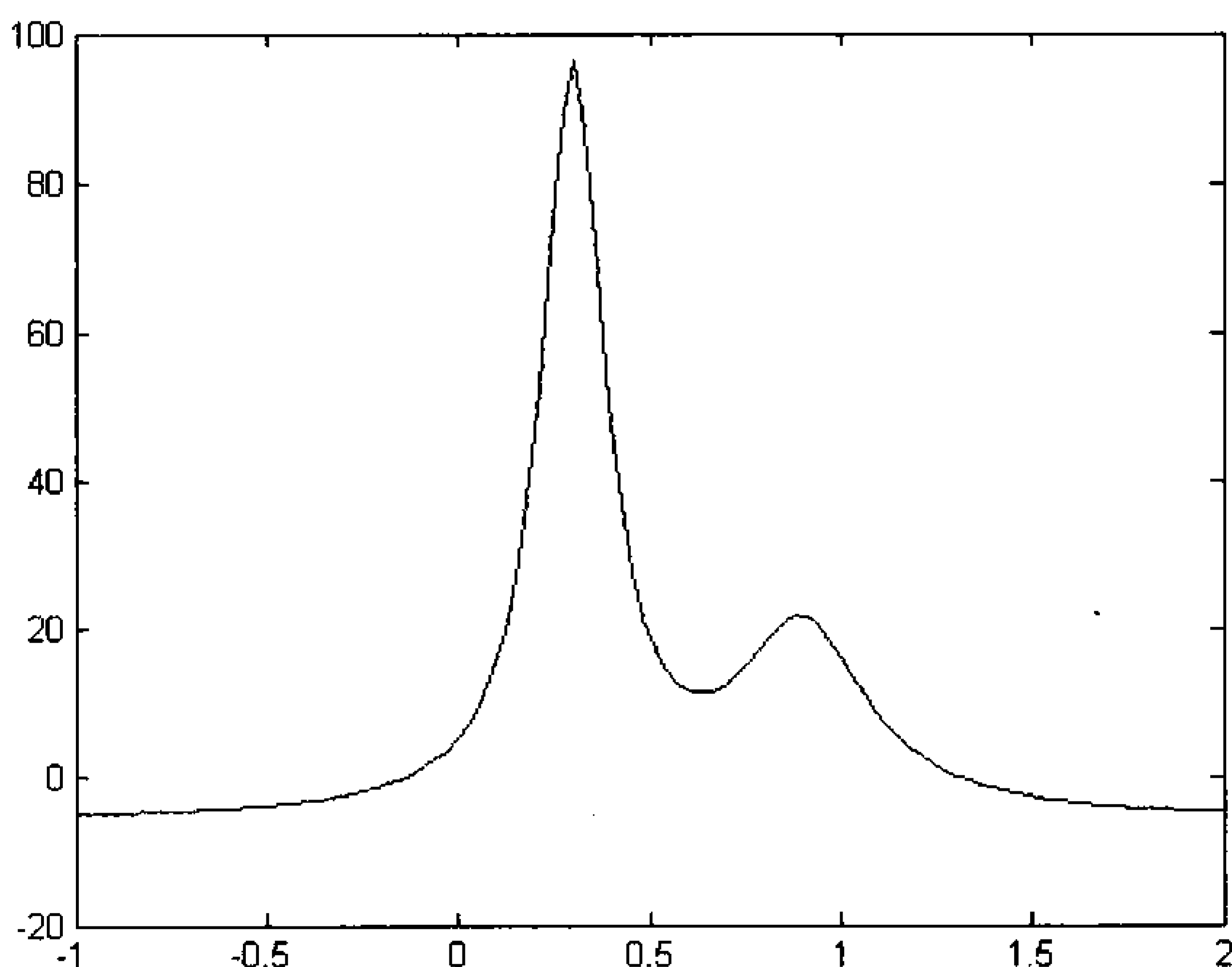


图 1-22　MATLAB 绘图示例

利用 MATLAB 进行三维绘图同样方便。下面的代码产生一条三维螺旋线，如图 1-23 所示。

```
>> t=linspace(0,10*pi);
>> plot3(sin(t),cos(t),t)
```

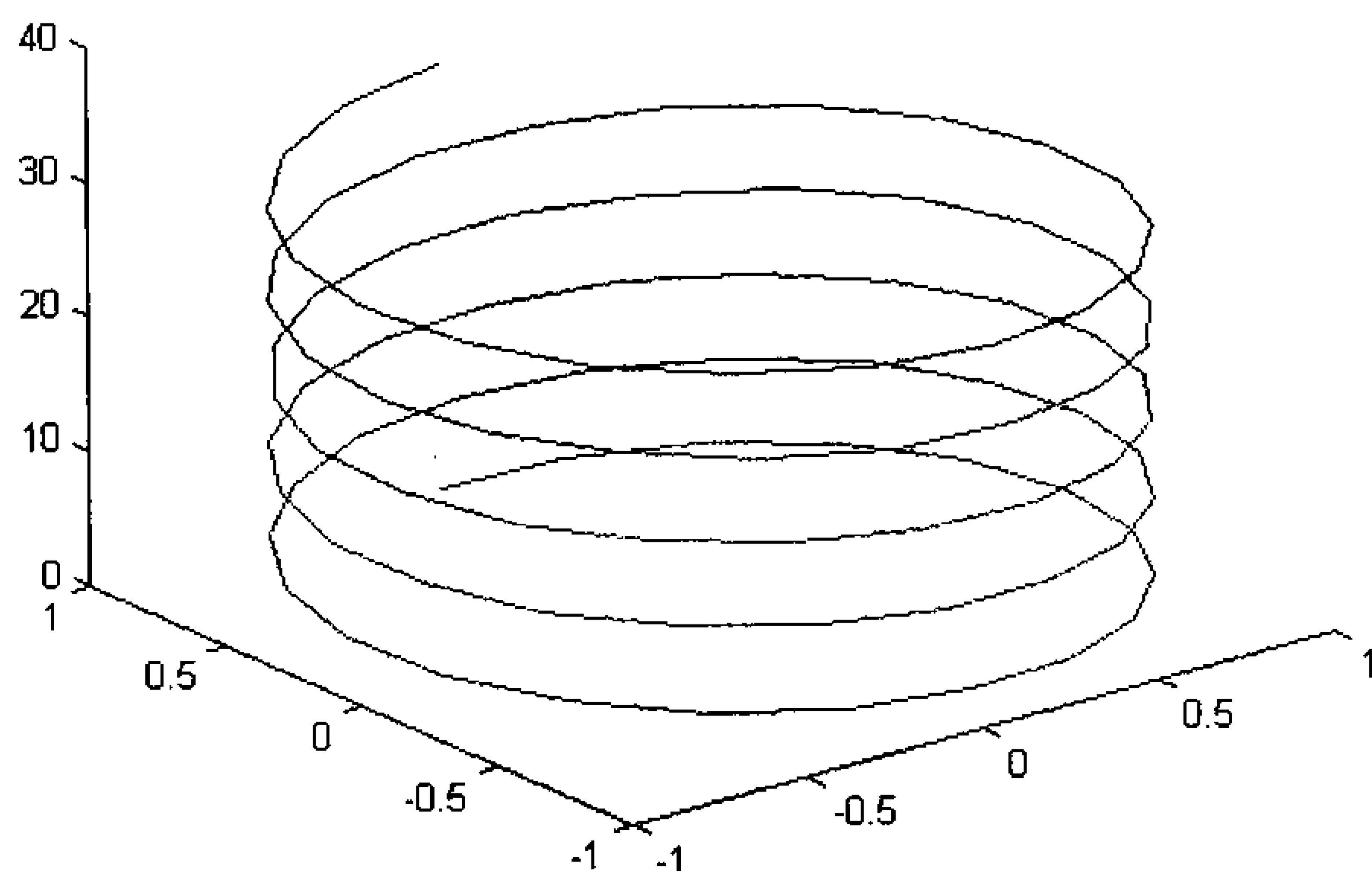


图 1-23　三维螺旋线

关于绘图更加详细的内容请参阅本书后面的章节。

1.3.3　M 文件的编写

MATLAB 不仅具有强大的数值处理和符号运算功能，而且可以像计算机高级语言一样进行程序设计。用 MATLAB 编程语言编写的程序称为 M 文件，它可以在 MATLAB 的工作空间运行。M 文件根据调用方式的不同分为命令文件和函数文件两类。命令文件不需要用户输入任何参数，也不会输出任何参数，它只是各种命令的叠加，有点像过去的 DOS 文件，运行时系统按顺序去执行文件中的各个语句。函数文件一般需要用户输入参数，也有可能输出用户需要的参数，在格式上函数文件必须以 function 语句作为引导，在功能上函数文件主要解决参数传递和调用的问题。在作用对象上，命令文件的作用对象是工作空间中的变量，因此，命令文件中的变量一般不需要预先定义；而函数文件中的变量是局部变量，除输入、输出的变量会驻留在工作空间以外，其他变量不会驻留在工作空间。

命令文件的编写很简单，通过执行【File】→【New】→【M-File】命令打开 M 文件编辑器，把想要执行的命令按行编写上去，编写完成后将文件确定一个名称保存起来即可。需要注意的是，命令文件存盘时不要忘记加上 M 文件的扩展名“.m”。当要执行时，只要在命令窗口的提示符号下输入该文件的文件名，按回车键后系统即可运行该命令文件。

函数文件一般分为定义行、帮助信息行、函数体和注释四部分。函数定义行为函数文件的第一行，功能是定义函数名、确定输入和输出变量。格式一般如下：

```
function<变量名>=函数名（参数）
```

紧跟定义行后以符号%开头的文字说明部分是帮助信息行。该行的文字信息在用户应用 lookfor+<关键字>或 help+<函数名>进行查询帮助信息时，系统显示该行的文字信息。接下来的是函数体，也就是函数实现其功能的程序，是函数文件编写的主要部分。在函数文件中凡是以%开头的文字部分均是注释内容，它可以被安排在程序的任何地方。

如例程 1.4，试编写一个命令文件，画出 $z=3-(x-3)^2-(y-3)^2$ 在 $x\in[0,6], y\in[0,6]$ 上的曲面。

例程 1.4　M 命令文件示例

```
%这是一个画二元函数 z=3-((x-3).^2+(y-3).^2)图的命令文件
D=[0:0.1:6];    %创建向量 D
[X,Y]=meshgrid(D);    %创建向量 X、Y 并赋值为 D
surf(X,Y,3-((X-3).^2+(Y-3).^2))    %绘制曲面图
axis off    %关闭坐标轴
```

得到的结果如图 1-24 所示。

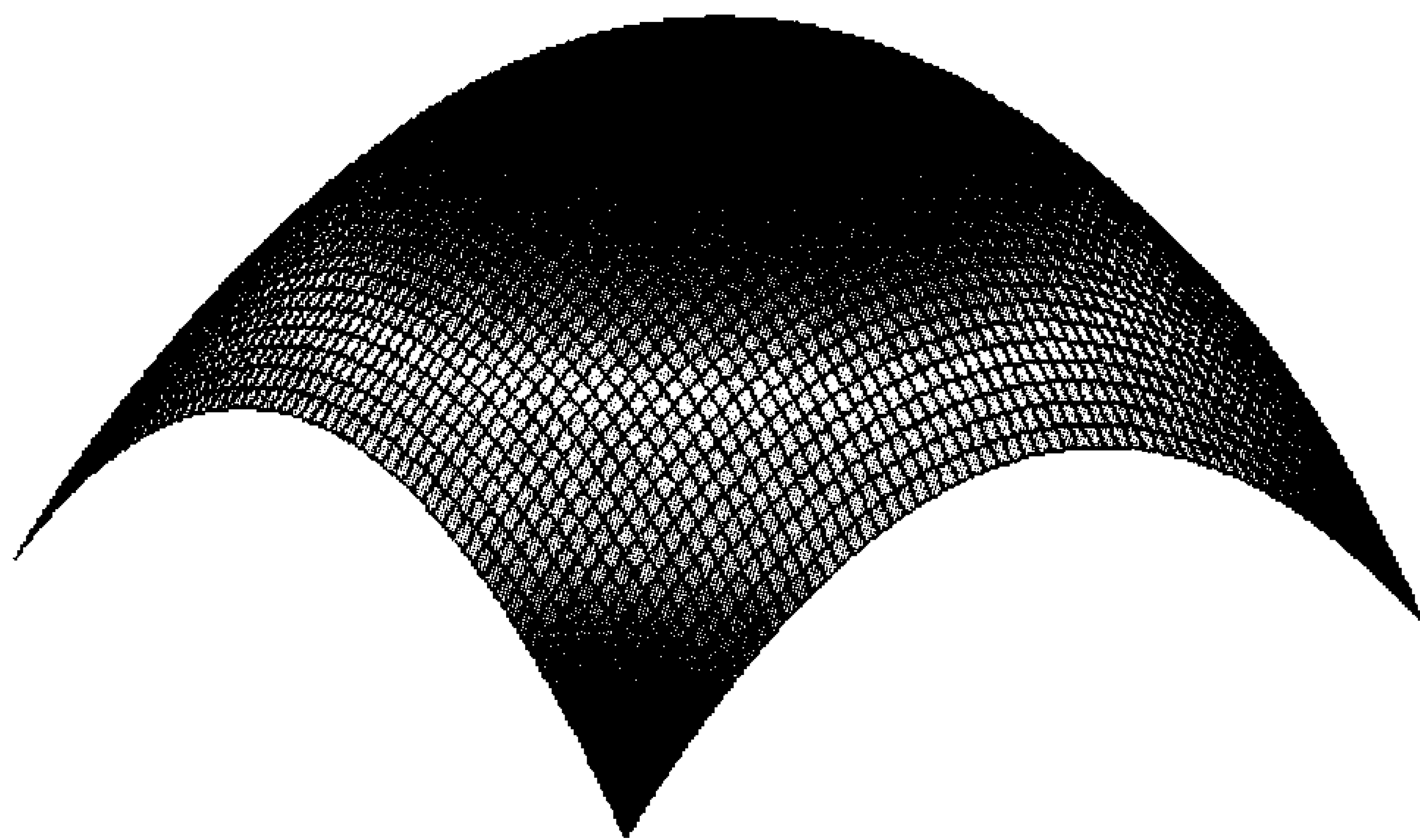


图 1-24　曲面图

例程 1.5 为编写一个 M 函数文件，求小于任何给定正整数的自然数的阶乘值。

例程 1.5　M 函数文件示例

```
function f=zsqf2(n)
%这是一个求小于任何给定正整数的自然数的阶乘值的示例
%调用格式 c=zsqf2(n)
%参数说明：n 可以是任意的正整数
f(1)=1
i=1;
while f(i)<ceil(n/i)
    f(i+1)=f(i)*(i+1)
    i=i+1;
end
```

在命令窗口调用这个函数，求得 10000 以内的自然数的阶乘值有 7 个，结果如下：

```
>>zsqf2(10000)
ans =
     1     2     6     24     120     720     5040
```

1.3.4　GUI 示例

GUI（Graphical User Interface）即图形用户界面，是由图形对象构建的用于人与计算机交互信息的界面。在图形用户界面上，用户可以根据界面上的提示信息完成自己的工作，而不需要记忆大量烦琐的命令，只需通过鼠标、键盘等简捷的方式与计算机交互信息、选

择想要运行的程序、控制程序的运行、实时显示图形信息。换言之，图形用户界面就是包含了各种图形控制对象（图形窗口、菜单、对话框和文本等）用于和计算机交互信息的图形界面。

例程 1.6 为建立一个进度条监视一个循环语句的进度。

例程 1.6　GUI 示例

```
h=waitbar(0,'请等待……');
for i=1:10000
    waitbar(i/10000)
end
```

建立的进度条如图 1-25 所示。

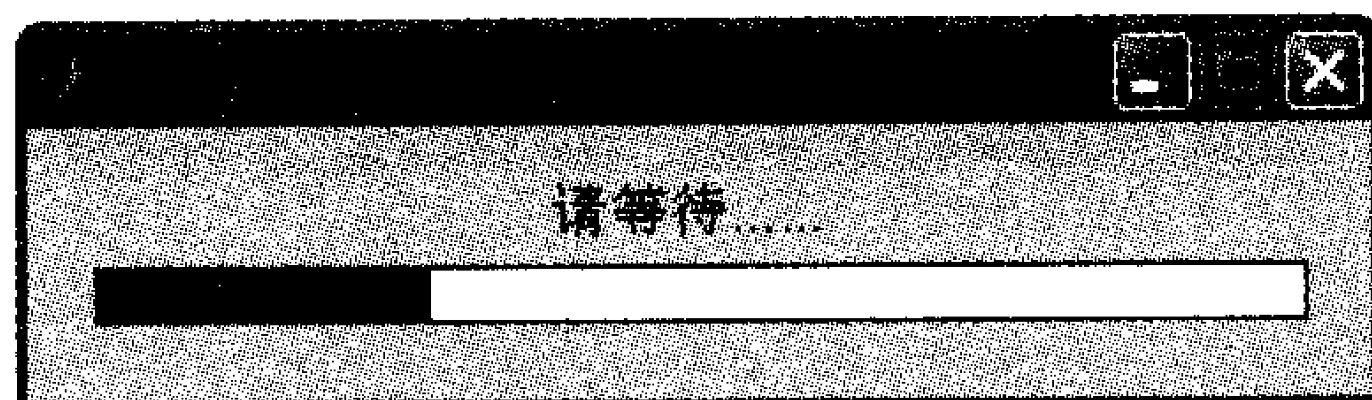


图 1-25　正在运行的进度条

关于 GUI 本书有专门的章节介绍，请读者参阅。

1.3.5　使用 Simulink 进行系统仿真

Simulink 是 MATLAB 最重要的组件之一，它提供一个动态系统建模、仿真和综合分析的集成环境。在该环境中，无须大量书写程序，而只需要通过简单直观的鼠标操作，就可构造出复杂的系统。Simulink 具有适应面广、结构和流程清晰及仿真精细、贴近实际、效率高、灵活等优点，并基于以上优点 Simulink 已被广泛应用于控制理论和数字信号处理的复杂仿真和设计。

下面的例子是用 Simulink 仿真生成一个正弦波发生器，如图 1-26 所示。

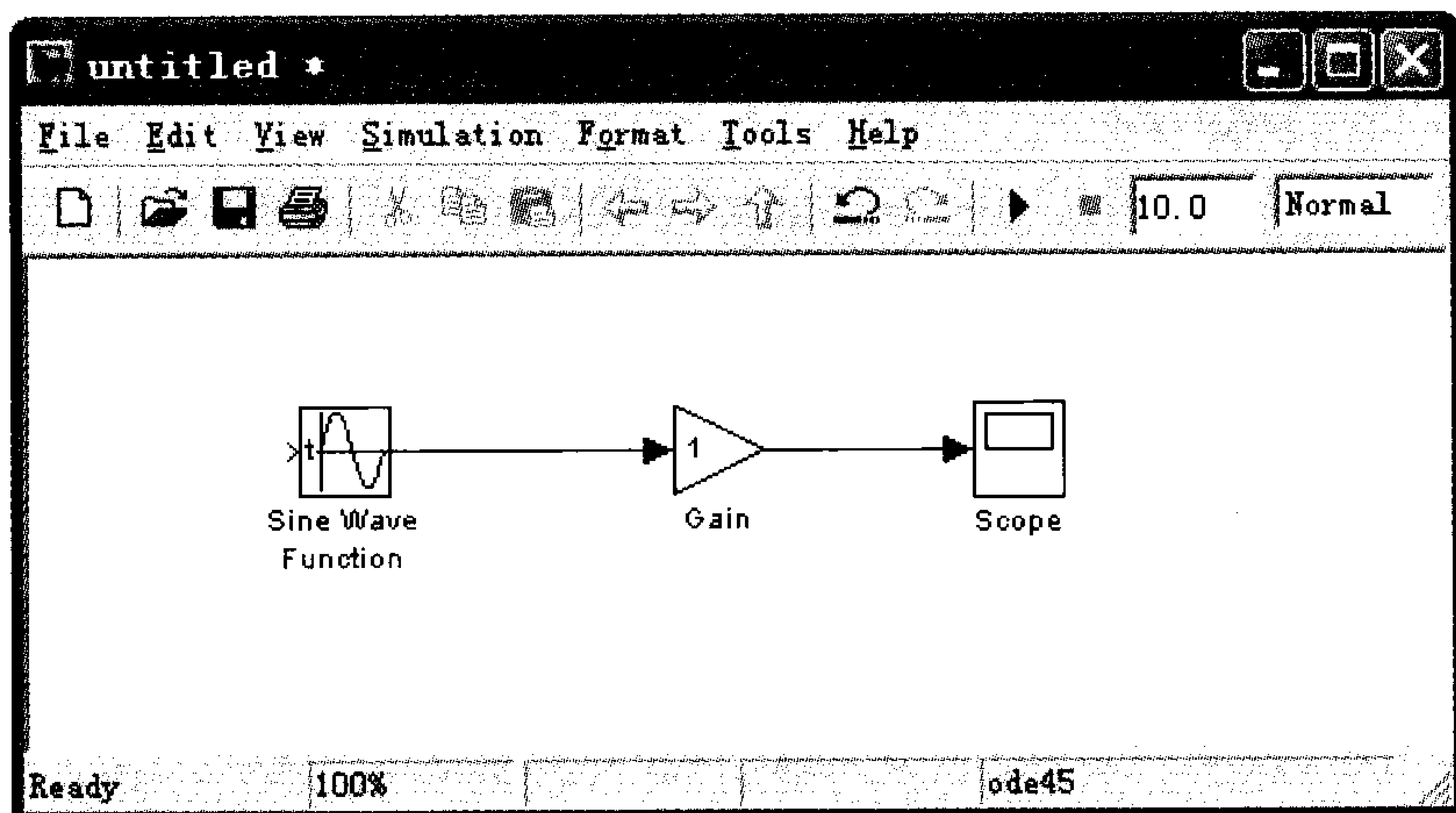


图 1-26　正弦波仿真系统

运行后可以得到如图 1-27 所示的结果。

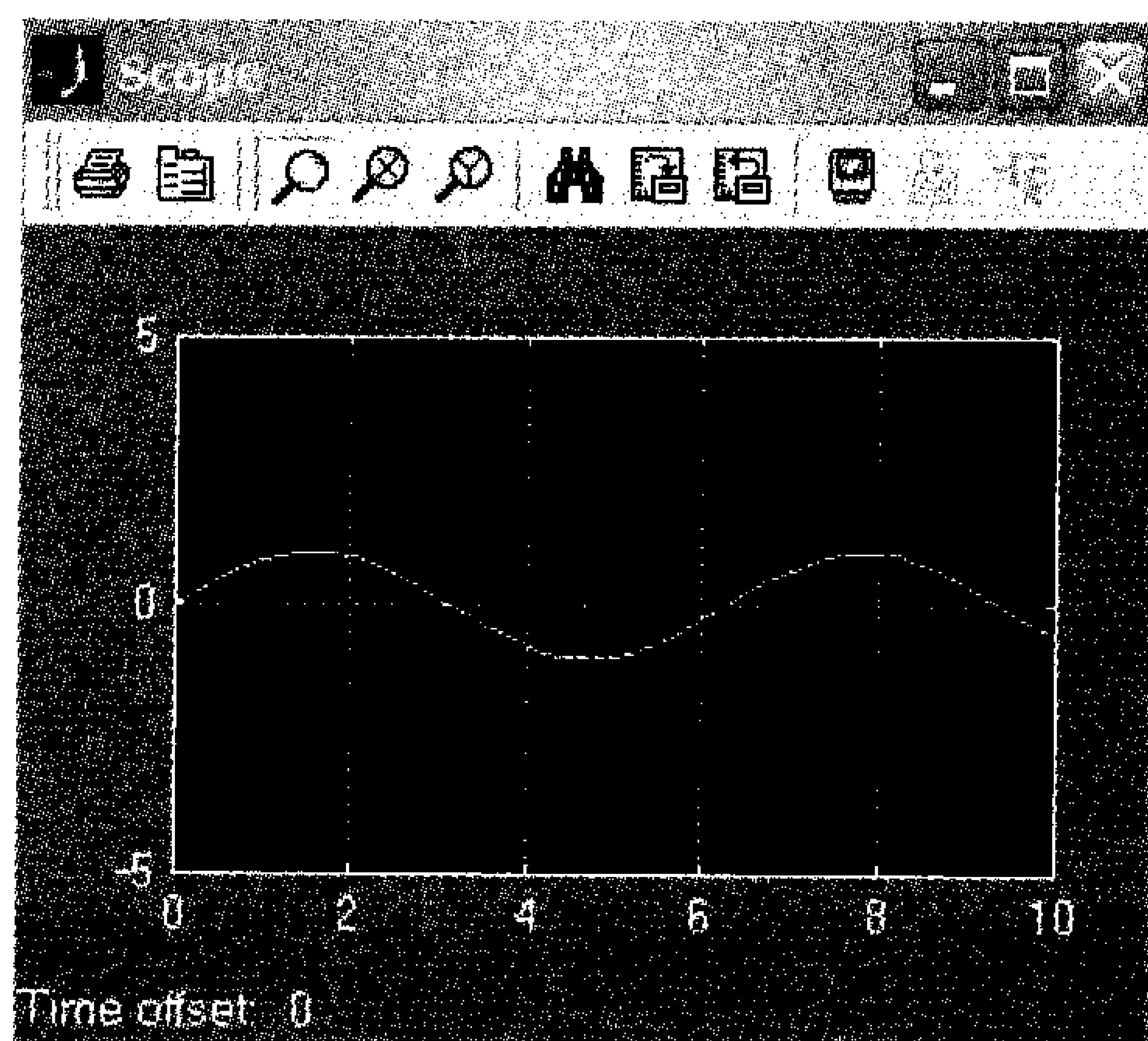


图 1-27　正弦波图形

1.4　MATLAB 学习技巧

学习 MATLAB 的过程中掌握一些技巧，并适当地加以运用，可以起到良好的学习效果。下面就简单介绍一下 MATLAB 学习的几个技巧。

1．要掌握好领域的专业知识

MATLAB 是一门较好的编程语言，简单易学，相对 VC 和 Delphi 等编程语言，门槛低了很多，但它归根结底只是算法的实现工具，解决问题最大的难点还在于如何针对实际问题，找到准确有效的解决方法，最好是设计好算法流程图，然后再考虑编程加以实现。在实际中，很多读者忽略了专业知识的学习和编程前期的准备，导致遇到问题时花费很多时间，比如调用工具箱函数时对函数本身的功能和机制缺乏深入的了解，输入输出的变量没有弄清楚，大大影响了解决问题的进程。

2．由浅入深、逐步深入

在进行 MATLAB 编程时，不一定要采取一步到位的方法，很多时候可以先在命令行工具里调用一两个命令查看一下效果，特别是一些核心的函数，看看能不能实现初步的功能，然后再考虑加入变量和其他辅助语句，程序逐步由小到大，这样程序出错的概率就小了很多。

3．习惯使用联机帮助

在学习 MATLAB 的过程中，要充分利用 MATLAB 帮助资源。MATLAB 帮助文档本身就是一本非常好的参考书。在帮助文档里不仅详细介绍了各个函数的用法，而且还可以引导用户养成非常好的编程风格。为了使 MATLAB 更容易上手，MATLAB 提供了非常丰富的 demo，用户只需在 MATLAB 的命令窗口输入 demo 即会出现非常多的范例，通过这些范例便能够更加清楚地查看 MATLAB 内部函数的功能和编写方式。

4. 多利用网上资源

学习 MATLAB 的读者可以充分利用网络来获取资源并加强交流。MATLAB 在我国已经拥有了许多用户，许多高校陆续开设了有关 MATLAB 的课程，清华大学、华中理工大学、中国科技大学等高校的 BBS 上还专门设立了 MATLAB 讨论区，另外还有很多专业的 MATLAB 网站和论坛。遇到疑难问题时，可以在相关的网上讨论、求助，也可以下载别人的源程序加以吸收学习，还可以通过在线交流工具得到同行的指点和点拨。

5. 要勤于尝试

对大部分人来说学习编程语言的目的绝对不是为了编程，而是要将其应用到实际工程中，解决实际问题。要敢于尝试，改变既有的程序来实现自己的某种想法，往往会有意想不到的良好效果，这样能加深对 MATLAB 和算法的理解，可以大大加快我们掌握它的进度。另外，MATLAB 语言编程效率比较高，很多时候并不太长的代码能实现非常复杂的功能。而在出现编译和运行错误时，往往只是某个赋值或者函数调用格式出现了问题，这时不要太着急，可以多次尝试不同的可能性，加以解决。

学习 MATLAB 的技巧很多，如一些好的调试方法、编译方法等，限于篇幅，这里不能一一列举，本书将在后面的章节中结合具体内容介绍一些技巧。但坦白地说，读者要掌握好这门工具，还是要结合本书内容，认真学习，打好基础，下足工夫，学习过程一定要有恒心、耐心和决心，制订切实可行的学习计划，提高编程的熟练程度，并将学到的知识在实际工程中加以利用，这样才能巩固所学知识，循序渐进地提高自己的 MATLAB 编程和应用水平。

第 2 章　MATLAB 数组和矩阵

数值计算是 MATLAB 中最重要、最有特色的功能之一。MATLAB 强大的数值计算功能使其成为诸多数学计算软件中的佼佼者，同时它也是 MATLAB 软件的基础。而数组和矩阵是数值计算的最基本运算单元，在 MATLAB 中，向量可以看做一维数组，而矩阵则可以看做二维数组。数组和矩阵在形式上没有区别，但二者的运算性质却有很大的不同，数组运算强调的是元素对元素的运算，而矩阵运算则采用线性代数的运算方式。

本章将介绍 MATLAB 数值计算中数组和矩阵的一些基础知识、数组的创建、寻访、赋值和运算，元胞数组和结构数组的一些使用方法，向量的创建和运算，矩阵的创建，矩阵的基本运算、特征参数运算、分解运算以及应用非常广泛的稀疏矩阵创建和应用。同时，将介绍矩阵运算在解决实际问题中的一些应用。读者通过本章可以逐渐体会到 MATLAB 强大的数值计算功能。

本章主要内容：

- 基础知识
- 数组及其运算
- 向量及其运算
- 矩阵运算及其应用

2.1　基础知识

MATLAB 的数值计算是以数组为基本单元的，而 MATLAB 数据类型的最大特点是每一种类型都以数组为基础。事实上，MATLAB 也是把每种类型的数据都作为数组来处理。这一节将介绍 MATLAB 中常用的一些数据类型，矩阵、数组的概念，常量和变量等基础知识。

2.1.1　数据类型

数据类型是掌握任何一门编程语言都必须首先了解的内容。MATLAB R2007 的数据类型主要有：逻辑、数值、字符串、矩阵、元胞、Java、函数句柄、稀疏以及结构等类型，其中数值型又有单精度型、双精度型以及整数型。而整数类型里面也分无符号型（uint8、uint16、uint32、uint64）和符号类型（int8、int16、int32、int64）两种，它们之间的层次关系如图 2-1 所示。在 MATLAB 中，所有的数据不管是属于什么类型，都是以数组或矩阵的形式保存的。

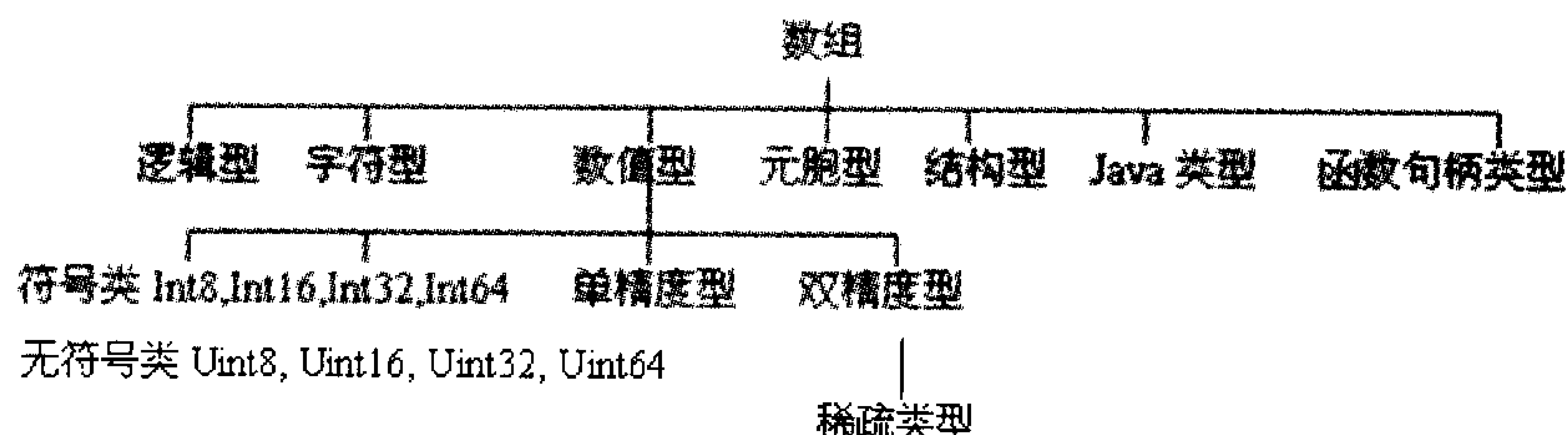


图 2-1　MATLAB 数据类型的层次结构图

1．**数值型数据**

数值类型包括整数型（带符号和无符号）和浮点数（单精度和双精度）两种。在默认状态下，MATLAB 将所有的数都看做是双精度的浮点数；双精度浮点数以 64 位存储，f：52 位，e：11 位，数的符号：1 位。指数以 e+1023 存储(1,211-2)，整个浮点数的分数部分不是 f，而是 1+f，IEEE 标准在 64 位中存储了 65 位的信息。所有的数值类型都支持基本的数组运算。除 int64 和 uint64 外所有的数值类型都可以应用于数学运算。

1）整型

如图 2-1 中所示，在 MATLAB 中有 4 种符号类型和 4 种无符号类型数据。符号类型可以表示正数、负数和零，但是它表示的数值的范围比无符号类型要小；相反，无符号类型只能表示非负数。

例 2.1

```
>> x = 325.499;
>> x = x + .001;
>> int16(x)          %对 x 取整
ans =
      326
>> intmax('int8')    %最大的整型类数值
ans =
      127
>> intmin('int8')    %最小的整型类数值
ans =
      -128
```

2）浮点型

MATLAB 中用双精度或单精度来表示浮点类型的数据，默认为双精度，但用户可以用一个简单的转换函数把任何数据用单精度来表示。

例 2.2

```
>> clear
>> x = single(25.783)    %用函数 single 创建单精度数据
x =
   25.7830
>> y=25.783              %创建的数据系统默认为双精度
y =
   25.7830
>> whos                  %用 whos 函数查看数据的类型，注意 x 与 y 的类型区别
  Name      Size            Bytes  Class     Attributes
```

```
  x           1x1                4  single
  y           1x1                8  double
```

3）复数

复数由两个独立部分组成：实部和虚部。虚部的基本单位为$\sqrt{-1}$，在 MATLAB 中常常用字母 i 或 j 来表示。

例 2.3

```
>> x = rand(3) * 5;  %复数的创建
>>y = rand(3) * -8;
>>z = complex(x, y)
z =
    4.0736 - 7.7191i   4.5669 - 7.6573i   1.3925 - 1.1351i
    4.5290 - 1.2609i   3.1618 - 3.8830i   2.7344 - 3.3741i
    0.6349 - 7.7647i   0.4877 - 6.4022i   4.7875 - 7.3259i
```

4）无穷大和 NaN

在 MATLAB 中，用特别的值 inf，-inf 和 NaN 分别表示正无穷大、负无穷大和不确定值。

例 2.4

```
>> x = log(0)        %负无穷
x =
   -Inf
>> y=1/0              %正无穷
y =
   Inf
>> clear
x = 7i/0              %不确定值
x =
   NaN +     Infi
>> whos
  Name        Size                Bytes  Class     Attributes
  x           1x1                    16  double    complex
```

5）数据类型的显示

最常见显示数据类型的函数为 whos，它可以显示变量的类型、大小、占用空间以及属性值，在 MATLAB 中还提供了其他以下函数来检验数据的类型，如表 2-1 所示。

表 2-1　常见数据类型识别函数

命　令	操　作
Whos x	显示 *x* 的数据类型
xType=class(x);	把 *x* 的数据类型赋值给标量
Isnumeric(x)	确定 *x* 是否为数值类型
isa(x,'integer'), isa(x,'uint64') isa(x,'float') ,isa(x, 'double') isa(x, 'single')	确定 *x* 是否为特别的数值类型（如任一整型、无符号 64-bit 整型、任意浮点型、双精度型、单精度型等）
isreal(x)	确定 *x* 是否为实数或复数类型
isnan(x)	确定 *x* 是否为非数
isinf(x)	确定 *x* 是否为无穷数
isfinite(x)	确定 *x* 是否为有限数

2．字符类型

在 MATLAB 中，字符串指的是一个统一编码的字符排列。字符串用一个向量或字符来表示，字符串存储为字符数组，每个元素占用一个 ASCII 字符，对于存储长度不一的字符串和包含多个串的数组最好使用元胞类型数组。

例 2.5

```
>> clear
>> name = 'Thomas R. Lee'          %创建 1 行 13 列的字符数组
name =
Thomas R. Lee
>> whos
  Name        Size            Bytes  Class     Attributes
  name        1x13               26  char
>> name = ['Thomas R. Lee     '; 'Senior Developer']     %创建二维字符数组
name =
Thomas R. Lee
Senior Developer
```

3．逻辑类型

逻辑数组类型是用数字 0 和 1 分别来表示逻辑假和逻辑真，逻辑类型的数据不一定是标量，MATLAB 也一样支持逻辑型数组，而且逻辑型的二维数组可能是稀疏的。

例 2.6

```
>> x = magic(4) >= 9      %创建逻辑型的数组
x =
     1     0     0     1
     0     1     1     0
     1     0     0     1
     0     1     1     0
```

4．元胞类型

元胞类型和结构类型是 MATLAB 中比较特殊的数据类型，元胞数组提供了不同类型数据的存储机制。元胞数组的每一个元素称为一个 Cell，每一个 Cell 自己本身又是一个数组。元胞类型组可以储存任意类型和任意维度的数组。用户可以通过与矩阵和数组中同样的矩阵索引的方法来存取数据，但表示方法有所不同，如用 A{1，2}来表示存取元胞数组的第 2 行、第 3 列的元胞。有关详细叙述请阅读 2.2 节中元胞数组。

5．结构类型

结构是包含已命名“数据容器”或域的数组。结构类型数组中的域可以包含任何类型的数据。正如标准的数组一样，结构继承了数组的有向性特点，用户可以构建任何有效类型的大小和形状的结构数组，包括多维结构类型数组。

例 2.7

```
>> patient.name = 'John Doe';                          %创建结构类型数组 patient
patient.billing = 127.00;
patient.test = [79 75 73; 180 178 177.5; 220 210 205];
>> patient
patient =
          name: 'John Doe'
```

```
billing: 127
test: [3x3 double]
```

6．Java 类型

MATLAB 提供的一个面向 Java 程序语言接口使用户能够从 Java 类中创建对象和调用有关这些对象的 Java 方法。用户可以从外部导入 Java 类型数据，也可以在 MATLAB 中创建自己的 Java 类型数组。本身和第三方的类都可以通过 MATLAB 的接口得到应用。

7．函数句柄类型

函数句柄用于间接调用一个函数的 MATLAB 值或数据类型。用户在调用其他函数时可以传递函数句柄，也可在数据结构中保存函数句柄以备用。

例 2.8

```
>> fhandle = @functionname        %用符号“@”构建函数句柄
fhandle =
    @functionname
sqr = @(x) x.^2                   %创建匿名函数的方法构建函数句柄
sqr =
@(x)x.^2
%-----------------------------------------------------------------
```

下面是一个简单函数句柄的例子。

例 2.9

```
在 MATLAB 的 M-file 编辑器中输入如下代码并以 plotFHandle 名字保存：
function x = plotFHandle(fhandle, data)
plot(data, fhandle(data))
然后在命令行窗口输入命令：
plotFHandle(@sin, -pi:0.01:pi)    %直接输入函数名 plotFHandle 来调用句柄函数
```

执行程序代码，得到的图形如图 2-2 所示。

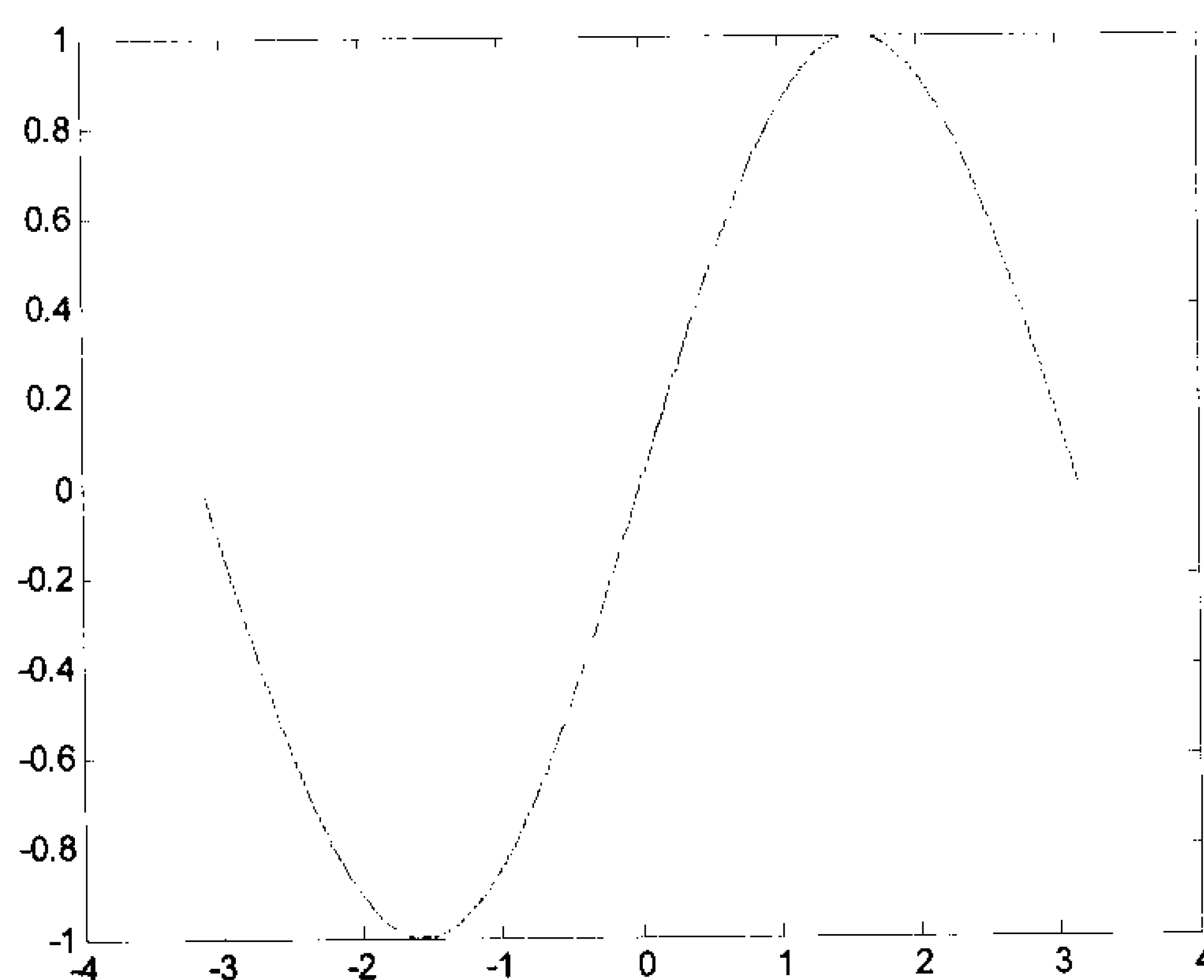


图 2-2 正弦函数图形

2.1.2 矩阵和数组的概念

在 MATLAB 的运算中，经常要使用到标量、向量、矩阵以及数组等概念。它们的含义

分别如下。

- 标量：指 1×1 的矩阵，它为只含有 1 个数的矩阵。
- 向量：在 MATLAB 中，指的是一维（1×n 或 n×1）的矩阵，它表示只有 1 行或 1 列的矩阵，通常在其他编程语言中看做一维数组。
- 矩阵：指一个二维的实数或复数数组，标量和向量都是它的特例。在很多非正式的场合，矩阵和数组是可以互换的，但严格地说，矩阵是一个表示线性转换的、二维的矩形实数或复数数组。
- 数组：指一组 n 维的矩形的实数或复数阵列（Array），n 可以为 1，表示一维的行或列；n 为 2 表示矩阵，n 可以为 3 或者更高的维数。

2.1.3 常量和变量

1. 常量

常量，在 MATLAB 中习惯称之为特殊变量，即系统自定义的变量，它们在 MATLAB 启动以后驻留在内存里面。在 MATLAB R2007 中常用的特殊变量如表 2-2 所示。

表 2-2 MATLAB 常用特殊变量表

特 殊 变 量	取 值
ans	MATLAB 中运行结果的默认变量名
pi	圆周率 π
eps	计算机中的最小数
flops	浮点运算数
inf	无穷大，如 1/0
NaN	不定值，如 0/0，∞/∞，0*∞
i 或 j	复数中的虚数单位，i=j= $\sqrt{-1}$
nargin	函数输入变量数目
narout	函数输出变量数目
realmin	最小的可用正实数
realmax	最大的可用正实数

在 MATLAB R2007 的命令窗口中输入一个表达式或者一组数据，系统将会自动把计算的结果赋值给“ans”变量。

例 2.10 在命令窗口计算 sin(2*π)。

```
>> pi
ans =
      3.1416
>> sin(2*pi)
ans =
    -2.4493e-016
```

2．变量

变量是任何程序设计语言的基本要素之一，MATLAB 也不例外。与常见的程序设计语言不同的是 MATLAB 并不要求事先对所使用的变量进行声明和变量类型的指定，MATLAB 语言会自动根据所赋予变量的值或对变量所进行的操作来识别变量的类型并分配合适的内存空间。如果赋值变量已存在时，MATLAB 将使用新值代替旧值，同时，以新值类型代替旧值类型。如下例所示。

```
Workload_teacher=5
```

创建一个 1×1 的名为 Workload_teacher 的矩阵，并将 5 作为元素的值。

注意 在 MATLAB 中，变量名的第一个字符必须是字母，后面可以跟个数不限的字符、数字或下画线；MATLAB 只使用变量名的前 63 个字符（MATLAB 6.5 版本以前是 31 个字符）；MATLAB 区分大小写，如 A 和 a 是不同的变量；要查看赋给任何变量的矩阵，只需要简单地输入变量名就可以了；关键字不能作为变量名，如 if，else 等。

- 变量的存储：在 MATLAB 中，存储当前工作空间中的变量，通过命令 Save 来完成。格式如下：

```
save                    %将所有变量存入文件 matlab.mat
save mydata             %将所有变量存入指定文件 mydata.mat
save mydata.mat         %将所有变量存入文件 mydata.mat
save 文件名 变量名列表   %存储指定的变量
```

- 变量的读取：MATLAB 提供 Load 命令来将数据文件中的变量载入当前工作空间。格式如下：

```
load mydata         %载入数据文件中的所有变量
load mydata A  x    %从数据文件中提取指定变量
```

- 工作空间变量的清除，格式如下：

```
clear          %清除当前工作空间中的所有变量
clear A  x     %清除指定的变量
```

2.1.4 数值计算应用的例子

下面将通过求曲线长度的例子来体会在 MATLAB 中如何使得复杂的求解数学积分问题过程变得简单、明了、快捷，也直观地感受一下 MATLAB 数值计算功能的强大。

例 2.11 求曲线长度。

曲线参数方程为：$\begin{cases} x(t)=\sin(2t) \\ y(t)=\cos(t), \quad 且\ t\in[0,3\pi] \\ z(t)=t \end{cases}$

解：根据曲线长度求法，我们可以列出求该曲线长度的表达式为：

$$f(t)=\int_0^{3\pi}\sqrt{4\cos(2t)^2+\sin(t)^2+1}\mathrm{d}t$$

```
>> t = 0:0.05:3*pi;                    %变量取值，步长为 0.05
```

```
plot3(sin(2*t),cos(t),t)                    %用 plot 函数画出曲线图形
function f = hcurve(t)                      %在 M-file 编辑器中编写如下代码，并保存
F = sqrt(4*cos(2*t).^2 + sin(t).^2 + 1);
>> len=quad(@hcurve,0,3*pi)                 %利用积分函数 quad()求出曲线长度
len =
    17.2220
```

所以，所求曲线的长度为 17.220，图形如图 2-3 所示。

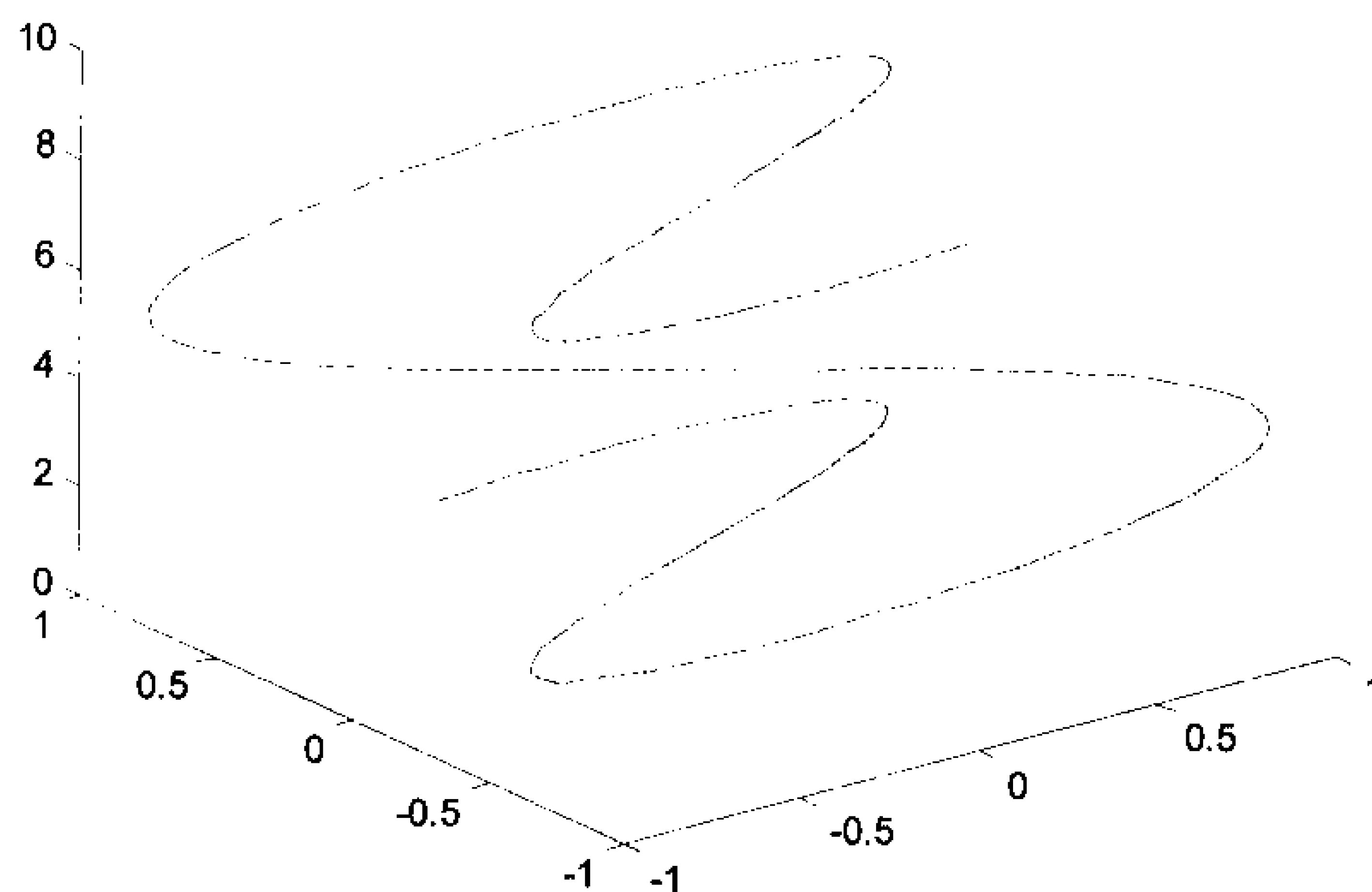


图 2-3　所求曲线的图形

2.2　数组及其运算

MATLAB 中最基本、最重要的运算就是数组运算和矩阵运算。在 MATAB 中，所有的运算都是以数组或者矩阵形式进行的。实际上，在线性代数以外，矩阵可以看做为二维的数值型数组。二维数组是由实数或复数排列而成的矩形构成的。在 MATLAB 中，从数据结构上看，数组和矩阵没有什么区别，但二者在运算性质上还是有比较大的区别。本节重点讨论数组运算的相关内容，矩阵及其运算将在 2.4 节中详细阐述。

2.2.1　数组的创建

首先看一个简单的例子。

例 2.12　算区间[-2π,2π]上的双曲正割值，并绘出图形。

```
>> x = -2*pi:0.01:2*pi;       %创建数组 x
>> y=sech(x);                 %创建数组 y
>> plot(x,y);
```

通过函数 plot 绘出这些数据的图形，如图 2-4 所示。

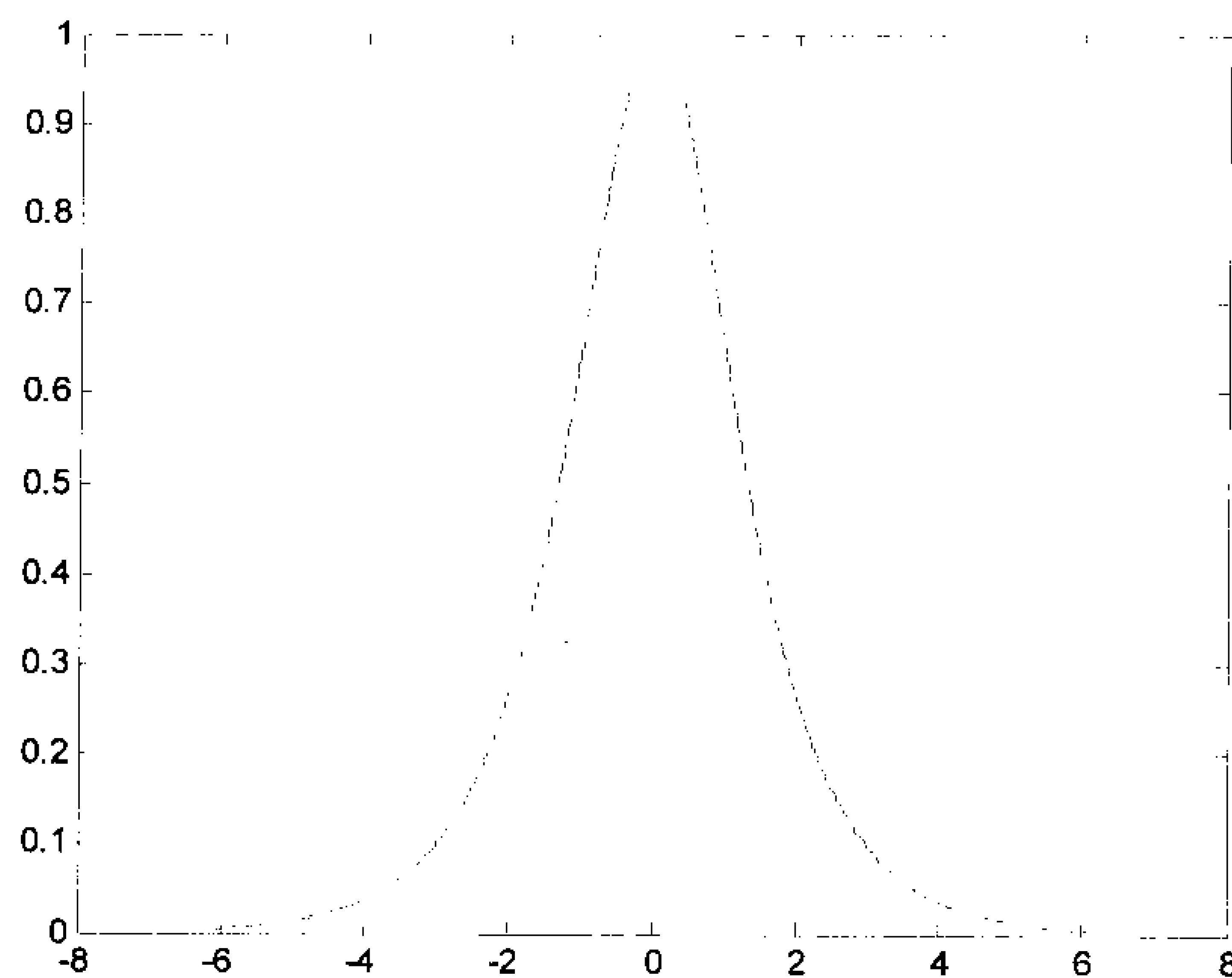


图 2-4 区间[0 2π]上正弦值图形

常见的数组创建方法有以下几种。

（1）直接输入。在 MATLAB R2007 命令窗口直接输入元素值。

例 2.13

```
>> x=[12 36 89;56 95 47]
x=
    12    36    89
    56    95    47
```

（2）增量法。格式：first:increment:last，表示创建的数组从 first 开始，至 last 结束，增量为 increment。

例 2.14

```
>>y=(0:0.25:pi)      % increment=1 时，可以省略
y =
  Columns 1 through 9
  0    0.2500    0.5000    0.7500    1.0000    1.2500    1.5000    1.7500    2.0000
  Columns 10 through 13
  2.2500    2.5000    2.7500    3.0000
```

（3）利用函数 linspace 或函数 logspace 创建数组。

格式：y = linspace(a,b,n)　%创建一个取值从 a 开始，至 b 结束，共有 n 个元素的数组

　　　y = logspace(a,b,n)　%创建一个取值从 10^a 开始，至 10^b 结束，共有 n 个元素的数组

例 2.15

```
>>z=linspace(0,100,10)   % linspace()函数直接定义了元素的个数，n 取值为 10
z =
  Columns 1 through 9
  0    11.1111    22.2222    33.3333    44.4444    55.5556    66.6667    77.7778    88.8889
  Column 10
  100.0000
>>z1=logspace(0,3,8)
z1 =
  1.0e+003 *
  0.0010    0.0027    0.0072    0.0193    0.0518    0.1389    0.3728    1.0000
```

（4）M 文件创建数组

在 MATLAB R2007 命令窗口输入 Edit 进入 M-file 编辑器，创建数组如图 2-5 所示，保存后退出。

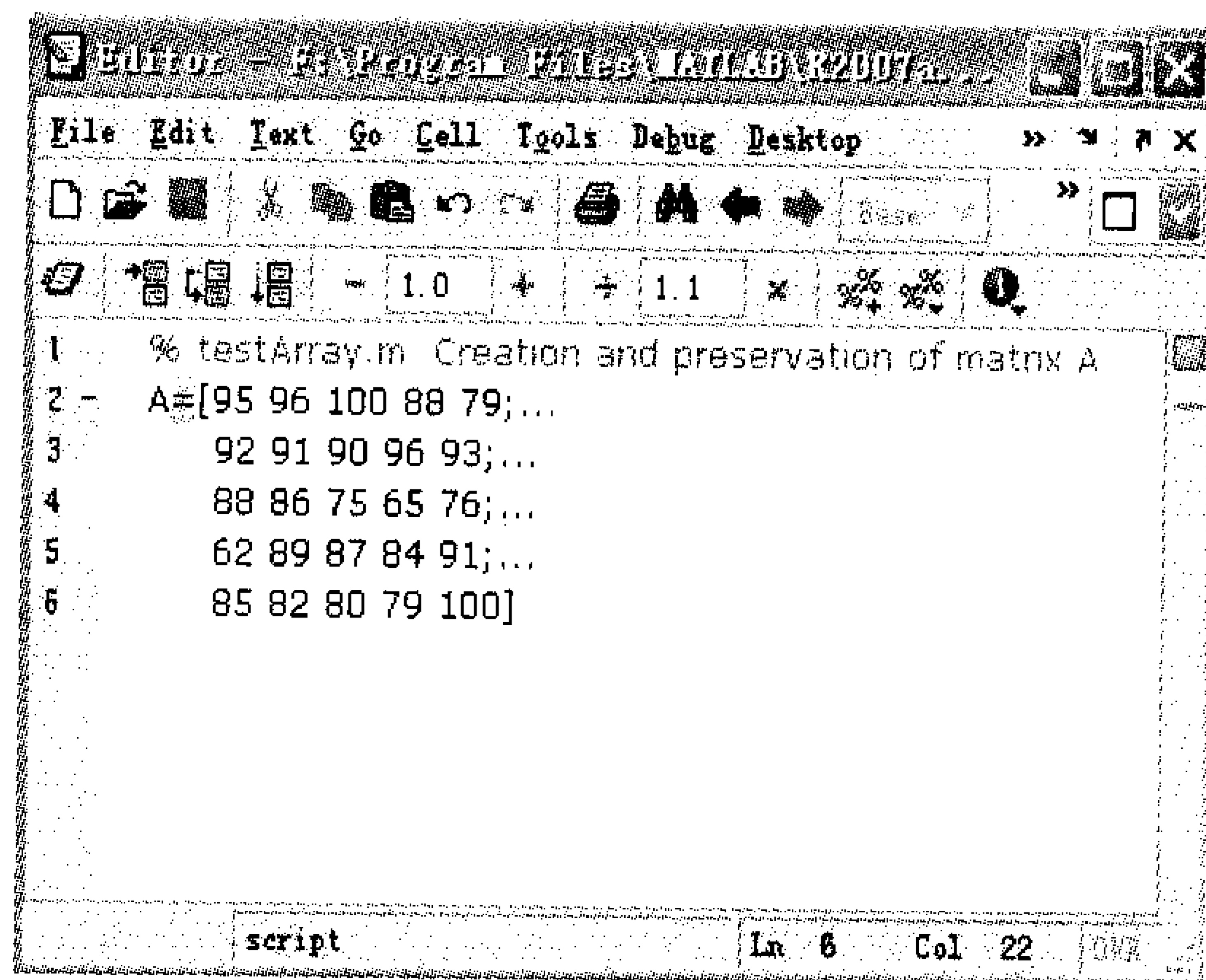


图 2-5　在 M-file 编辑器中编辑数组

然后在命令行窗口输入 M 文件名 testarray 用来显示刚刚创建的数组。

```
>> testarray
A =
    95    96   100    88    79
    92    91    90    96    93
    88    86    75    65    76
    62    89    87    84    91
    85    82    80    79   100
```

2.2.2　数组的寻访和赋值

1．数组的寻访

1）一维数组的寻访

寻访什么子数组依据数组的下标而定，即 x（index）中的 index，且 index 可以是单个正整数或正整数数组，但其取值必须在区间[1,end]范围内。

例 2.16　一维数组的子数组的寻访。

```
>>Clear
>> x=[15 22 33 94 85 77 60]
x =
    15    22    33    94    85    77    60
>> x(6)              %直接寻访一维数组的第 6 个元素
ans =
    77
>> x([1 3 5])       %寻访一维数组的多个元素
```

```
ans =
      15    33    85
>> x(4:end)          %寻访数组中第 4 个至最后 1 个元素
ans =
      94    85    77    60
>> x(find(x>70))   %寻访附加条件的数组中的元素
ans =
      94    85    77
>> x(2:5)             %寻访数组的第 2 个至第 5 个元素
ans =
      22    33    94    85
```

2）二维数组的子数组寻访

二维数组的子数组寻访类似于一维数组，常用的二维数组的子数组寻访指令如表 2-3 所示。

表 2-3　常用二维数组的子数组寻访和赋值指令

子数组寻访	功 能 说 明
A(i,j)	寻访数组的第 *i* 行第 *j* 列元素
A(i,:)	寻访数组的第 *i* 行的所有元素
A(:,j)	寻访数组的第 *j* 列的所有元素
A(:)	寻访数组中的所有元素，数组中元素按从上到下，从左到右的次序排列成单列数组
A(s)	寻访的数组由 *s* 指定，生成一个新的一维数组
A(L)	寻访的数组由逻辑数组 *L* 指定，生成一个列数组
A(i,j)=S	把数组 *S* 中元素的值赋给 *A*(*i*,*j*)，注意 *S* 的行和列的维度必须与 *A*(*i*,*j*)的一致
A(:)=D(:)	把数组 *D* 中的所有元素赋值给 *A*，*A* 中行和列的维度不变，但 *A* 与 *D* 中的元素个数必须相等；二者行和列的维度不一定相同
A(s)=S	对 *A* 中部分元素重新赋值，*A* 中行和列的维度不变。数组中 *s* 的长度必须与一维数组 *S* 一致

例 2.17　二维数组的子数组的寻访。

```
>>Clear
>> A=magic(5)
A =
      17    24     1     8    15
      23     5     7    14    16
       4     6    13    20    22
      10    12    19    21     3
      11    18    25     2     9
>> A(3,5)            %寻访数组第 3 行、第 5 列的元素
ans =
      22
>> A([4 7 9 20 23])  %寻访数组的第 4、7、9、20、23 个元素，数组元素是按列储存的
ans =
      10     5    12     2    22
```

```
>> A(2:4,4)            %寻访数组的第 4 列的第 2 行至第 4 行元素
ans =
    14
    20
    21
>> s=[1 5 8];
>> A(s)
ans =
    17    11     6
```

2．数组的赋值

例 2.18

```
>> clear
>> x=[15 22 33 94 85 77 60];
>> x(8)=25
x =
    15    22    33    94    85    77    60    25    %为数组 x 添加第 8 个元素
>> A=magic(5);
>> s=[1 5 8];
>> S=[80 90 95];     %将数组 S 中的元素分别赋值给矩阵中第 1、5、8 个元素
>> A(s)=S
A =
    80    24     1     8    15
    23     5     7    14    16
     4    95    13    20    22
    10    12    19    21     3
    90    18    25     2     9
>>  A(6,7)=33        %当数据保存在数组的元素之外的时候数组将会自动增大空间来保存这个新增的元素
A =                  %魔方矩阵由 5×5 变成了 6×7
    80    24     1     8    15     0     0
    23     5     7    14    16     0     0
     4    95    13    20    22     0     0
    10    12    19    21     3     0     0
    90    18    25     2     9     0     0
     0     0     0     0     0     0    33
>> A(1:6,3)=1          %把数组第 3 列元素全部赋值为 1
A =
    80    24     1     8    15     0     0
    23     5     1    14    16     0     0
     4    95     1    20    22     0     0
    10    12     1    21     3     0     0
    90    18     1     2     9     0     0
     0     0     1     0     0     0    33
```

2.2.3 数组运算

数组运算是 MATLAB 软件中所定义的规则，其目的就是为了更加方便地管理数据、操作更加简单、指令形式更加直观和更有效地执行计算。前面已经讲到数组和矩阵在形式上

是相同的，但在运算上却有很大的不同。这一小节将重点阐述数组运算，有关它与矩阵运算的区别将在2.4节中进行详细阐述。

1）数组与常数（标量）的四则运算

数组中的每一个元素和标量进行加、减、乘、除运算。

例 2.19

```
>> x=[1 3 8;2 6 5;7 9 1];
>> y=3*x./2+8
y =
    9.5000   12.5000   20.0000
   11.0000   17.0000   15.5000
   18.5000   21.5000    9.5000
```

2）数组之间的四则运算

数组之间的四则运算为数组中的元素与元素之间的加、减、乘、除运算。

例 2.20

```
>> x=[45 36 12;11 25 48;21 78 66];
>> y=[9 6 4;11 5 8;7 3 22];
>> z=x+y
z =
    54    42    16
    22    30    56
    28    81    88
>>z1=z./x
z1 =
    1.2000    1.1667    1.3333
    2.0000    1.2000    1.1667
    1.3333    1.0385    1.3333
```



在数组的四则运算中做四则运算时两个数组必须有相同的维数，否则MATLAB将会提示出错信息（如例2.21）；除法运算符号“./”和“.\”的计算结果是不同的，关系如下：a./b=b.\a。

例 2.21

```
>> x=[1 6 8]
x =
   1     6     8
>> y=[5 8 9 7]
y =
   5     8     9     7
>> z=x./y
??? Error using ==> rdivide
Matrix dimensions must agree.
```

3）数组的幂运算

注意，数组的幂运算和矩阵的幂运算是完全不同的，前者运算符号为“.^”，是元素对元素的幂运算，而后者运算符号为“^”，请读者注意下例中两种运算的区别。

例 2.22

```
>> x=[1 3 8;2 6 5;7 9 1];
>> y1=x.^2              %数组的幂运算
y1 =
     1     9    64
     4    36    25
    49    81     1
>> y2=x^2               %矩阵的幂运算
y2 =
    63    93    31
    49    87    51
    32    84   102
```

4）数组的指数运算

在 MATLAB R2007 中，提供了函数 exp 来进行指数运算。

例 2.23

```
>> x=[1 3 8;2 6 5;7 9 1];
>> y1=exp(x)
y1 =
  1.0e+003 *
    0.0027    0.0201    2.9810
    0.0074    0.4034    0.1484
    1.0966    8.1031    0.0027
```

5）数组的对数运算以及开方运算

MATLAB 提供的数组对数运算和开方运算的函数分别为 log 和 sqrt。

例 2.24

```
>> x=[1 3 8;2 6 5;7 9 1];
>> y2=log(x)
y2 =
         0    1.0986    2.0794
    0.6931    1.7918    1.6094
    1.9459    2.1972         0
>> y3=sqrt(x)
y3 =
    1.0000    1.7321    2.8284
    1.4142    2.4495    2.2361
    2.6458    3.0000    1.0000
```



在执行数组运算的时候，参与运算的数组必须有相同的维数，且得到的结果是一个与原数组大小相同的数组；在求数组的乘法、除法、乘方、三角以及指数运算的时候，它与矩阵有着明显的区别。详细内容请参见表 2-4 及 2.4 节的相关内容。

2.2.4 元胞数组

元胞数组和结构数组是 MATLAB 中比较特殊的数据类型，元胞数组的每一个元素称为一个 Cell，每一个 Cell 自己本身又是一个数组。而且，元胞数组中的各元胞的内容和维数

可以不相同。结构数组中的元素也可以是不同的数据类型，但不同于元胞数组的是：结构数组的引用是通过属性名来实现的。下面分别介绍元胞数组和结构数组。

元胞数组的各个元素为元胞（Cell），每一个元胞作为一个单元。它可以存放各种不同类型的数据，如数组、矩阵、字符串、元胞数组以及结构数组等，且各个元胞的内容可以是不相同的。

1）创建元胞数组

常见的有两种方法：语句直接生成和由各元胞创建。

例 2.25 用直接法生成元胞数组。

```
>> A={'test',[5 6;7 8],{'You are welcome','Thanks'},rand(3)}
A =
    'test'    [2x2 double]    {1x2 cell}    [3x3 double]
>> whos
  Name       Size              Bytes  Class    Attributes
  A          1x4                 514  cell
```

例 2.25 中得到了一个四元胞的数组，*A*（1，1）为字符串，*A*（1，2）为 2×2 的矩阵，*A*（2，1）为 1×2d 元胞数组，*A*（2，2）为 3×3 的矩阵。

例 2.26 用各元胞创建方法生成元胞数组。

```
>> B{1,1}=[1,2,3;4,5,4;7,8,9];
>> B{1,2}='kitty';
>> B{2,1}=3+7i;
>> B{2,2}=eye(3);
>> B
B =
              [3x3 double]      'kitty'
    [3.0000 + 7.0000i]      [3x3 double]
>> whos
  Name       Size              Bytes  Class    Attributes
  A          1x4                 514  cell
  B          2x2                 410  cell
```

2）显示元胞数组的内容

在 MATLAB 中显示元胞数组的内容可以通过如例 2.26 中在 MATLAB 命令窗口输入元胞数组名 B 来显示，还可以通过函数 celldisp 来显示，如例 2.27。

例 2.27

```
>> celldisp(B)
 B{1,1} =
      1     2     3
      4     5     4
      7     8     9
B{2,1} =
    3.0000 + 7.0000i
B{1,2} =
    kitty
B{2,2} =
      1     0     0
      0     1     0
      0     0     1
```

同时，我们还可以通过函数 cellplot 用图形的方式直观显示出元胞数组的内容，如图 2-6 所示。

```
>> cellplot(B.'legend')     %添加显示参数 legend 使得元胞数组内容更加清晰、直观
```

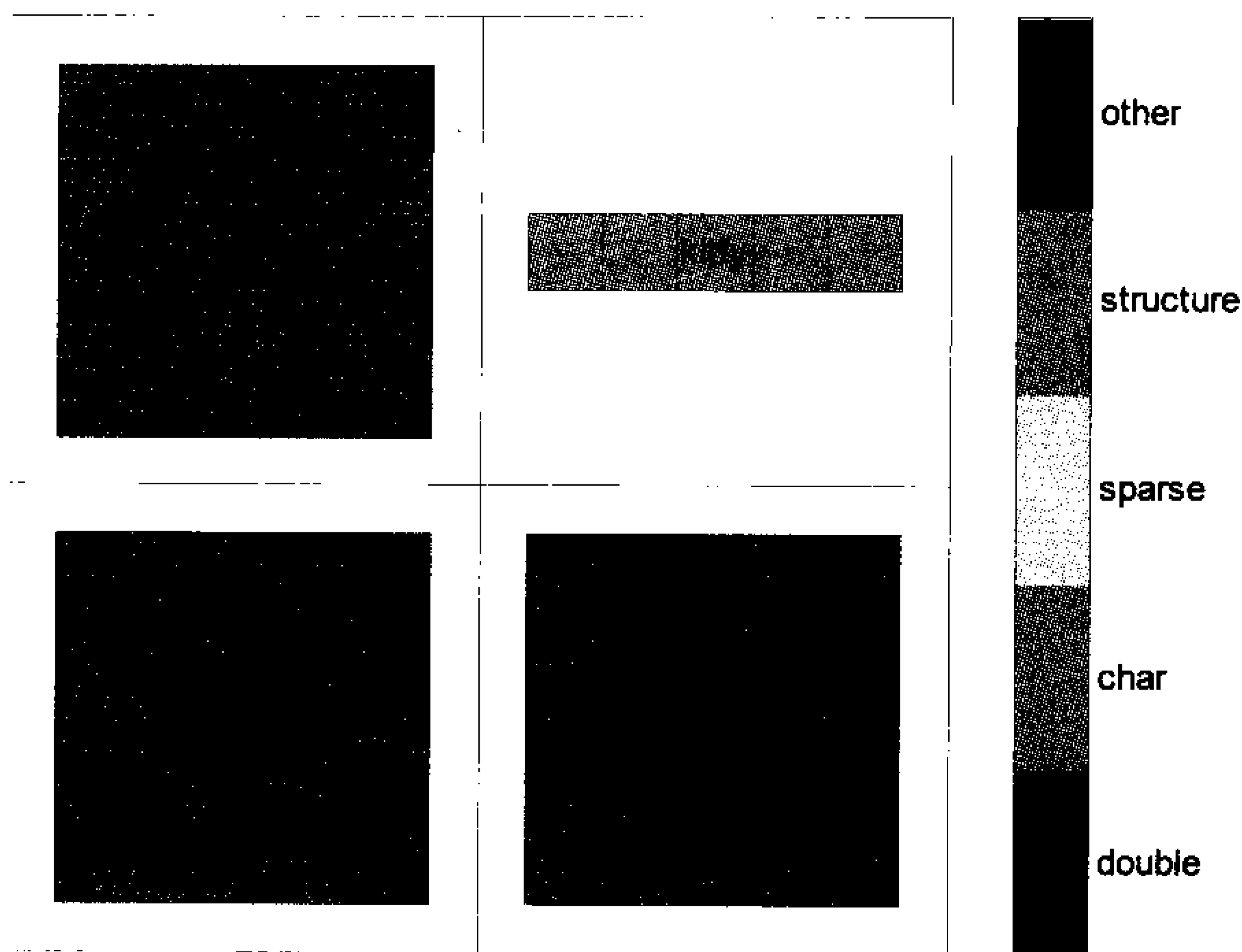


图 2-6 图形方式显示元胞数组内容

3）元胞数组和数值数组之间的转换

在 MATLAB 中，常常通过 num2cell、cell2mat 等函数实现元胞数组和数值数组之间的转换，下面通过例 2.28 来体会这几个函数的用法。

例 2.28

```
>> A=magic(3);
>> C=num2cell(A,2)  %魔方矩阵矩阵转换成元胞数组
C =
    [1x3 double]
    [1x3 double]
    [1x3 double]
>> celldisp(C)          %显示元胞数组的内容
C{1} =
      8     1     6
C{2} =
      3     5     7
C{3} =
      4     9     2
%--------------------------------------------------------------------------------
>> C = {[1] [2 3 4]; [5; 9] [6 7 8; 10 11 12]}    %元胞数组转换成数值数组
C =
    [           1]    [1x3 double]
    [2x1 double]    [2x3 double]
>> C{1,1}
ans =
     1
>> C{2,1}
ans =
```

```
        5
        9
>> M = cell2mat(C)
M =
        1       2       3       4
        5       6       7       8
        9      10      11      12
```

2.2.5 结构数组

在 MATLAB 中，结构数组和元胞数组都可以存放不同数据类型的数据。只不过结构数组在内容上更加丰富，在应用上更加广泛。比如，图形对象就包含很多如 Tag、Name、Color、Position 以及 String 等不同类型的属性，且每一个图形对象都具有这些属性，而往往每个属性的属性值却不尽相同，这时就可以用结构数组存储图形对象的属性值。

结构数组由结构（Structure）组成，且每一个结构都包含多个结构域（Fields）。例如，多个图形对象构成一个结构数组，一个图形对象就是一个结构，一个对象的属性就是一个域。数据（如图形对象的属性值）不能够直接存储在结构中，只能够存放在结构域中，结构域中可以存放任何类型和任何大小的数组。

1）结构数组的创建

常见的结构数组的创建也有两种方法：语句直接创建和通过函数 Struct 创建。

例 2.29 用直接法创建存放图形对象属性的结构数组。

```
>> h(1).tag='openbutton';
>> h(1).name='open';
>> h(1).color='green';
>> h(1).position=[ 245 110 10 20];
>> h                  %h 为结构数组，h(1)为结构数组中的结构，而 tag，name 等为结构域
h =
1x2 struct array with fields:
     tag
     name
     color
     position
>> h(1)
ans =
          tag: 'openbutton'
         name: 'open'
        color: 'green'
     position: [245 110 10 20]
```

例 2.30 通过 Struct 函数创建存放图形对象属性的结构数组。

```
>> clear
>> h(2)=struct('tag','openbutton','name','open','color','green');
>> h(3)=struct('tag','exitbutton','name','exit','color','red');
>> h                                  %利用 struct 函数把图形对象的属性值存储在 h 结构数组中
h =
1x3 struct array with fields:
     tag
```

```
        name
        color
```

2）结构数组的数据获取和设置

例 2.31　获取例 2.30 结构数组中的数据，并设置结构数组中的数据。

```
>> x1=h(2).tag                          %利用‘.’获取结构数组中的数据
x1 =
openbutton
>> x2=h(3).color
x2 =
red
>> x3=getfield(h,{2},'name')            %通过函数 getffield 获得结构数组中的数据
x3 =
open
%-----------------------------------------------------------------------------------------------
>>h(2)=struct('tag','openbutton','name','open','color','green');
>> h=setfield(h,{2},'color','black')    %通过函数 setffield 设置结构数组中的数据
>> h(2).color
ans =
black
```

3）结构数组的其他操作

例 2.32　添加结构数组中的结构域。

```
>>clear
>> h(2)=struct('tag','openbutton','name','open','color','green');
>> h(2).position=[225 188 10 25];       %添加结构域
>> h(2)
ans =
          tag: 'openbutton'
         name: 'open'
        color: 'green'
     position: [225 188 10 25]
```

例 2.33　删除结构数组中的结构域。

```
>> clear
>> h(2)=struct('tag','openbutton','name','open','color','green');
>> h=rmfield(h,'color')          % 调用 rmfield 函数删除结构数组中的结构域 color
h =
1x2 struct array with fields:
    tag
    name
```

例 2.34　结构数组的运算。

```
>> student(1)=struct('No',07332688,'name','David','score',[89 90 78;85 70 98])
student =
          No: 7332688
        name: 'David'
score: [2x3 double
>> aver=mean(student(1).score)    %调用函数 mean 计算结构数组 student 的平均分数
aver =
    87      80      88
```

2.3 向量及其运算

向量是组成矩阵的基本元素之一，我们可以把它看做成一维数组。MATLAB 中同样提供功能强大的向量运算。这一节将介绍向量的基本知识、向量的基本运算、向量的点积、叉积和混合积运算。

2.3.1 向量的创建

向量的创建方法和数组的创建方法类似，也可以通过命令窗口直接输入。

例 2.35 在命令行直接创建向量。

```
>> a1=[1 2 3 4 5 6]   % 创建一个行向量
a1 =
     1     2     3     4     5     6
>> a2=[7;8;9]             %创建一个列向量
a2 =
     7
     8
     9
>> a3=a2'                %通过转置操作符号“'”实现行向量和列向量之间的转换
a3 =
     7     8     9
```

当输入的向量元素数量比较多时，可以通过函数 linspace 或运算符“:”来创建向量，但是通过这两种方法创建的向量元素之间有着一定的规律。

例 2.36 创建等差元素向量。

```
>> b1=0:2:20              %等差步长 n=2
b1 =
     0     2     4     6     8    10    12    14    16    18    20
>> b2=linspace(0,10,5)   %在[0 10]区间生成 5 个元素
b2 =
          0    2.5000    5.0000    7.5000   10.0000
```

2.3.2 向量的基本运算

1）向量的四则运算

（1）向量与常数的运算。向量中每一个元素与常数做加、减、乘、除运算。

例 2.37

```
>> b2=linspace(0,10,5)
b2 =
     0    2.5000    5.0000    7.5000   10.0000
>> b3=(b2*5+2)/5
b3 =
     0.4000    2.9000    5.4000    7.9000    10.400
```

（2）向量之间的加法（减法）运算。向量中每个元素与另一个向量中相对应的元素进

行加法（减法）运算。

例 2.38

```
>> a1=[1 2 3 4 5 6]
a1 =
     1     2     3     4     5     6
>> b1=0:2:20
b1 =
     0     2     4     6     8    10    12    14    16    18    20
>> c1=a1+b1                    %向量之间的操作同样必须有相同的维度
??? Error using ==> plus
Matrix dimensions must agree.
>> b2=2:2:12
b2 =
     2     4     6     8    10    12
>> c1=b2+a1
c1 =
     3     6     9    12    15    18
```

2）向量的点积运算

在高等数学中，两个向量的点积等于一个向量的模与另一个向量在这个向量方向上的投影的乘积，MATLAB R2007 提供了函数 dot 来进行点积运算。在 MATLAB 中，向量点积运算的时候应注意两个向量的维度要保持一致。

例 2.39

```
>> x1=[2 9 8 7]
x1 =
     2     9     8     7
>> x2=[6 5 1 4]
x2 =
     6     5     1     4
>> y=dot(x1,x2)
y =
    93
%------------------------------------------------------------------------------------------
>> x3=[12 20 30 18 5]
x3 =
    12    20    30    18     5
>> y=dot(x1,x3)
??? Error using ==> dot at 30
A and B must be same size.
```

所以，当两个向量的维数不一致的时候将出现例 2.39 中的错误。

3）向量的叉积运算

在高等数学中，两个向量的叉积为两个向量的交点，并与此两向量所在平面垂直的向量，MATLAB R2007 提供了函数 cross 进行叉积运算。

例 2.40

```
>> x1=[2 9 8 7];
>> x2=[6 5 1 4];
>> y1=cross(x1,x2)
??? Error using ==> cross at 37
```

```
A and B must have at least one dimension of length 3.
```

从例 2.40 可以看出，和向量的点积运算一样，向量的叉积运算的两个向量维度也要求一致，而且必须是 3。

```
>> x1=[2 9 8];
>> x2=[6 5 1];
>> y1=cross(x1,x2)
y1 =
    -31    46    -44
```

4）向量的混合积运算

混合积的几何意义为：它的绝对值表示以向量为棱的平行六面体的体积。

混合积运算是通过函数 dot 和 cross 一起来完成的。在 MATLAB 中，做混合积运算的时候应该注意两个运算的先后顺序不能颠倒，看下面的例子。

例 2.41

```
>> x1=[1 5 8];
>> x2=[3 6 9];
>> x3=[2 4 7];
>> y=dot(x3,cross(x1,x2))
y =
    -9
>> y= cross(x3,dot(x1,x2))          %把点积和叉积运算颠倒将会出错
??? Error using ==> cross at 31
A and B must be same size.
```

2.4　矩阵运算及其应用

MATLAB 语言是由最初专门用于矩阵运算的计算机语言发展而来的。MATLAB 语言最重要、最基本的功能就是进行实数或复数矩阵的运算，其所有的数值计算功能都是以矩阵（或数组）为基本单元来实现的，尤其是在 MATLAB 图形图像处理、信号处理和控制理论等方面涉及大量的矩阵运算。矩阵运算和数组运算在形式上有很多相似之处，但是在 MATLAB 中二者的运算性质是不同的，数组运算强调的是元素对元素的运算，而矩阵运算则采用线性代数的运算方式。读者不能把二者混为一谈，以免产生一些不可预期的错误。这一节将重点阐述矩阵及其运算和矩阵在数值计算中的应用。如表 2-4 所示列出了矩阵和数组的常用运算操作。

表 2-4　常用的数组运算和矩阵运算操作

数组运算		矩阵运算	
操　作	功能描述	操　作	功能描述
A+B	对应的元素相加	A+B	同数组运算
A-B	对应的元素相减	A–B	同数组运算
S.*B	标量 S 分别与 B 中元素的积	S*B	同数组运算
A.*B	对应的元素相乘	A*B	内维相同的矩阵相乘
S./A	A 中元素左除 S	S/A	矩阵 A 左除 S

（续表）

数组运算		矩阵运算	
操　作	功能描述	操　作	功能描述
A./B	B 中元素左除 A 中元素	A/B	矩阵 A 右除矩阵 B
B.\A	上行 A 的另一种表达形式	B\A	矩阵 B 左除矩阵 A（与上一行意义不同）
A.^S	S 为整数时，A 中元素自乘 S 次	A^S	S 为整数时，且 A 为方阵则自乘 S 次
A.^S	S 为小数时，对 A 中各元素分别求非整数幂	A^S	S 为小数时，方阵 A 非整数乘方
S.^A	分别以 A 中元素为指数求幂值	S^A	A 为方阵时，标量 S 的矩阵乘方
A.’	非共轭转置	A’	矩阵的共轭转置
Exp(A)	以 e 为底数，分别以 A 中元素为指数求幂值	Expm(A)	求矩阵 A 的指数的函数
Log(A)	对 A 中各元素求对数	Logm(A)	求矩阵 A 的对数的函数
Sqrt(A)	对 A 中各元素求平方根	Sqrtm(A)	求矩阵 A 的平方根的函数
F(A)	求 A 各个元素的函数值	Funm(A)	矩阵的函数运算（A 必须为方阵）

其中，A、B 为矩阵，S 为标量。

2.4.1 矩阵的创建

矩阵的创建方法和数组的创建方法类似，也可以通过直接输入、增量法、利用函数 linspace 或 logspace 等几种方式，这里不再赘述。当创建矩阵的数据比较多的时候，可以通过 MATLAB 提供的矩阵编辑器（Matrix Editor）来生成或者修改矩阵。

在 MATLAB 主窗口中执行【Window】→【Workspace】命令，在打开的“Workspace”窗口（如图 2-7 所示）中显示了当前创建的一些矩阵变量。双击其中的一个变量弹出“Array Editor–A”窗口（如图 2-8 所示），则可以修改矩阵中的数据或者新建矩阵数据。

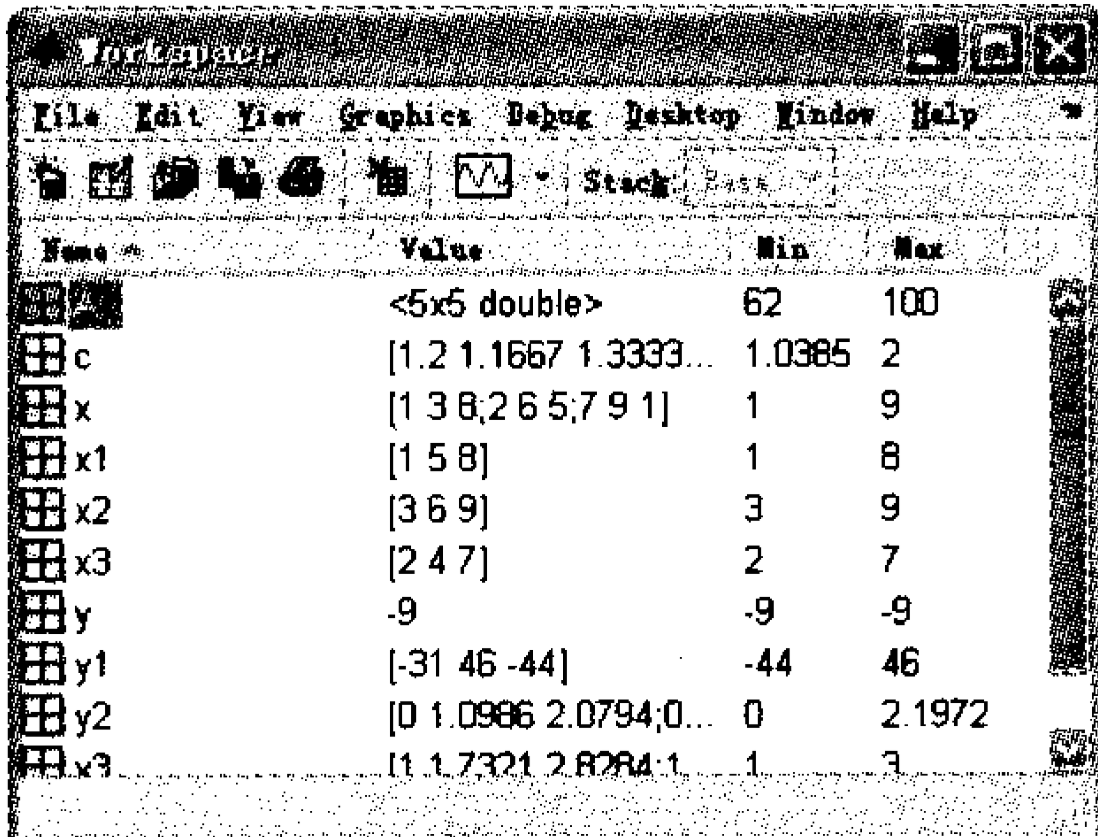


图 2-7 “Workspace”窗口

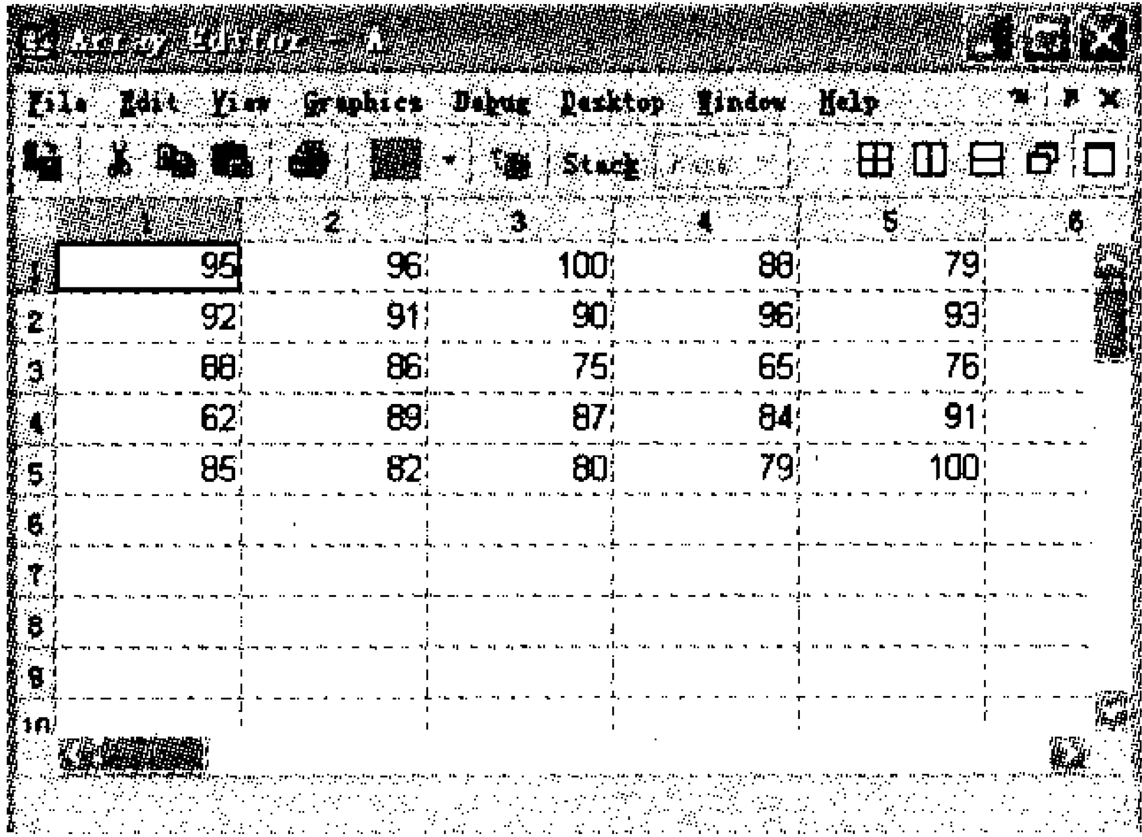


图 2-8 “Array Editor–A”窗口

还有一个是在数组创建方法中没有介绍的，那就是采用特殊矩阵函数来创建矩阵；另外，还可以用从 M 文件中把外部数据导入矩阵的方法创建矩阵。

2.4.2 矩阵的基本操作

1）元素的提取

在矩阵操作中常常要获取矩阵中的某个特殊元素，在MATLAB中可以通过A（row，column）或者A（*n*）来提取单个元素，获取矩阵的部分元素可以采用冒号运算符方法，具体如下。

A(:)为A的所有元素。

A(:,:)为二维矩阵A的所有元素。

A(:,k)为A的第*k*列，A(k,:)为A的第*k*行。

A(k:m)为A的第*k*个到第*m*个元素。

A(:,k:m)为A的第*k*列到第*m*列组成的子矩阵。

例2.42 对于范德蒙矩阵A有：

```
>> a=[1 2 3];
>> A=vander(a)
A =
     1     1     1
     4     2     1
     9     3     1
>> A(3,1)                    %获得范德蒙矩阵A的第3行、第1列处的元素
ans =
     9
>> A(5)    %其中n为索引，因为矩阵中的元素是按列存储的，即1 4 9 1 2 3 1 1 1这样的序列
ans =
     2
```

例2.43 获取矩阵A的多个元素。

```
>> A=hilb(3)
A =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000
>> S=A(1,3)+A(2,3)+A(3,3)
S =
    0.7833
>> S=sum(A(1:3,3))            %提取矩阵的多个元素
S =
    0.7833
>> S=sum(A(:))                %求所有元素之和
S =
    3.7000
```

2）矩阵的旋转

矩阵的旋转操作在矩阵运算中也是经常用到的。MATLAB提供的旋转操作主要有以下3个函数。

- fliplr(A)：矩阵的左右旋转。
- flipud(A)：矩阵的上下旋转。

- rot90(A)：矩阵逆时针旋转 90°，rot90(A,k)为逆时针旋转 $k\times90°$。

例 2.44 求矩阵 A 的旋转矩阵。

```
>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2
>> B=fliplr(A)
B =
     6     1     8
     7     5     3
     2     9     4
>> C=flipud(A)
C =
     4     9     2
     3     5     7
     8     1     6
>> D=rot90(A,2)     %逆时针旋转两个 90°
D =
     2     9     4
     7     5     3
     6     1     8
```

3）矩阵的转置

矩阵的转置在控制理论等问题里面使用比较广泛，矩阵的转置与数组的转置操作是不相同的，在 MATLAB 中提供矩阵转置操作符为 ‘’'，而数组转置操作符为 “.’”。

例 2.45 试比较矩阵的转置与数组的转置。

```
>> A=[1 2;1i 2i];
>> B=A'                  %矩阵转置
B =
   1.0000                  0 - 1.0000i
   2.0000                  0 - 2.0000i
>> C=A.'
C =                      %数组转置
   1.0000                  0 + 1.0000i
   2.0000                  0 + 2.0000i
```

4）矩阵的缩放

（1）矩阵的扩大。当将数据保存在矩阵的元素之外时，矩阵将会自动增大空间来保存这个新增的元素。

例 2.46

```
>> A=eye(3)
A =
     1     0     0
     0     1     0
     0     0     1
>> A(3,5)=8
A =
     1     0     0     0     0
```

```
      0     1     0     0     0
      0     0     1     0     8
```

（2）矩阵的缩小。我们可以通过将行或列指定为空数组[]，从而删除矩阵中的行或列。但是不能从矩阵中删除单个的元素。

例 2.47

```
>> A=rand(3)
A =
    0.7143    0.8328    0.5758
    0.4076    0.5191    0.2587
    0.8901    0.0236    0.2069
>> A(1,3)=[]               % 从矩阵中删除单个的元素系统给出错误信息
??? Subscripted assignment dimension mismatch.
>> A(:,2)=[]
A =
    0.7143    0.5758
    0.4076    0.2587
    0.8901    0.2069
```

5）获取矩阵的信息

在矩阵数值运算的时候常常要涉及矩阵的相关信息，MATLAB 提供了表 2-5 中常用的几个函数来获取矩阵的信息。

表 2-5 常用的获取矩阵信息函数

函 数 名	功 能 说 明
Length	返回矩阵最长的维的长度
Ndims	返回矩阵的维数
Numel	返回矩阵的元素个数
Size	返回矩阵的每一维的长度

```
>> A=rand(3);
>> length(A)
ans =
     3
>> size(A)
ans =
     3     3
>> numel(A)
ans =
     9
```

2.4.3 特殊矩阵

在矩阵论中介绍过许多特殊矩阵，这里将介绍一些在 MATLAB 中常用到的特殊矩阵以及稀疏矩阵，因为在控制理论的问题中经常要用到这些矩阵。

1）零矩阵和全 1 矩阵

MATLAB 提供 zeros( )和 ones( )两个函数分别生成这两个矩阵。

例 2.48

```
>> testarray
A =
    95    96   100    88    79
    92    91    90    96    93
    88    86    75    65    76
    62    89    87    84    91
    85    82    80    79   100
>> B=zeros(size(A))                          %生成一个大小与矩阵 A 相同的零矩阵
B =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
>> D=ones(4,4)    或者  D=ones(4)            %生成一个 4 行 4 列的全 1 矩阵
D =
     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1
```

2）单位矩阵

单位矩阵是特殊矩阵中用得最多的矩阵之一，一般用 I 来表示单位矩阵，MATLAB 提供的函数为 eye 。

例 2.49 求单位矩阵。

```
>> I=eye(3,4)
I =
     1     0     0     0
     0     1     0     0
     0     0     1     0
>> I=eye(size(B))
I =
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

3）其他一些特殊矩阵如表 2-6 所示。

表 2-6 特殊矩阵列表

矩阵名	函数名	矩阵特点	例子	
			示例输入	显示结果
对角矩阵	Diag()	对角线上为任意数，其他元素为 0	a=[1 2 6] diag(a)	1 0 0 0 2 0 0 0 6
上三角阵	Triu()	对角线的下面部分全为 0	a=[1 2 3;4 5 6;7 8 9] triu(a)	1 2 3 0 5 6 0 0 9
下三角阵	Tril()	对角线的上面部分全为 0	a=[1 2 3;4 5 6;7 8 9] tril(a)	1 0 0 4 5 0 7 8 9

（续表）

矩阵名	函数名	矩阵特点	例子	
			示例输入	显示结果
随机矩阵	Rand()	由随机数组成的矩阵	Rand(3)	0.9649 0.9572 0.1419 0.1576 0.4854 0.4218 0.9706 0.8003 0.9157
魔方矩阵	Magic()	每行、每列以及对角线上的元素之和相等	Magic(3)	8 1 6 3 5 7 4 9 2
范德蒙阵	Vander()	呈现范德蒙式特点	a=[1 2 3] vander(a)	1 1 1 4 2 1 9 3 1
伴随矩阵	Compan()	产生一个矩阵的伴随矩阵	a=[1 2 8 7] compan(a)	−2 −8 −7 1 0 0 0 1 0
Hilbert 阵	Hilb()	又称为病态矩阵，*i* 行、*j* 列的元素均为 1/*i*+*j*-1	a=hilb(3)	1.0000 0.5000 0.3333 0.5000 0.3333 0.2500 0.3333 0.2500 0.2000
Hadamard	Hadamard()	元素均由 1 或−1 组成，且满足 H'*N=N*I 条件。 该矩阵在数值分析、组合数学、信号处理应用广泛	Hadamard(4)	1 1 1 1 1 −1 1 −1 1 1 −1 −1 1 −1 −1 1

4）特殊矩阵应用举例

（1）随机矩阵和零矩阵应用举例。

例 2.50 Monte Carlo 评估方法。

```
>> close all;
rand('seed',.123456)
NumberInside = 0;
PiEstimate = zeros(500,1);    %通过 zeros 函数生成一个 500 行、1 列的零矩阵赋值给 PiEstimate
for k=1:500
    x = -1+2*rand(100,1);
    y = -1+2*rand(100,1);
    NumberInside = NumberInside + sum(x.^2 + y.^2 <= 1);
    PiEstimate(k) = (NumberInside/(k*100))*4;
end
plot(PiEstimate)
title(sprintf('Monte Carlo Estimate of Pi = %5.3f',PiEstimate(500)));
xlabel('Hundreds of Trials')
```

执行程序结果如图 2-9 所示。

（2）魔方矩阵是特殊矩阵中最常用的矩阵之一，例 2.51 即为运用魔方矩阵函数 magic 产生的三维图形应用实例。

例 2.51 在命令行窗口输入如下代码。

```
>>for n = 8:11
subplot(2,2,n-7)
surf(magic(n))
title(num2str(n))
axis off
view(30,45)
axis tight
end
```

执行程序结果如图 2-10 所示。

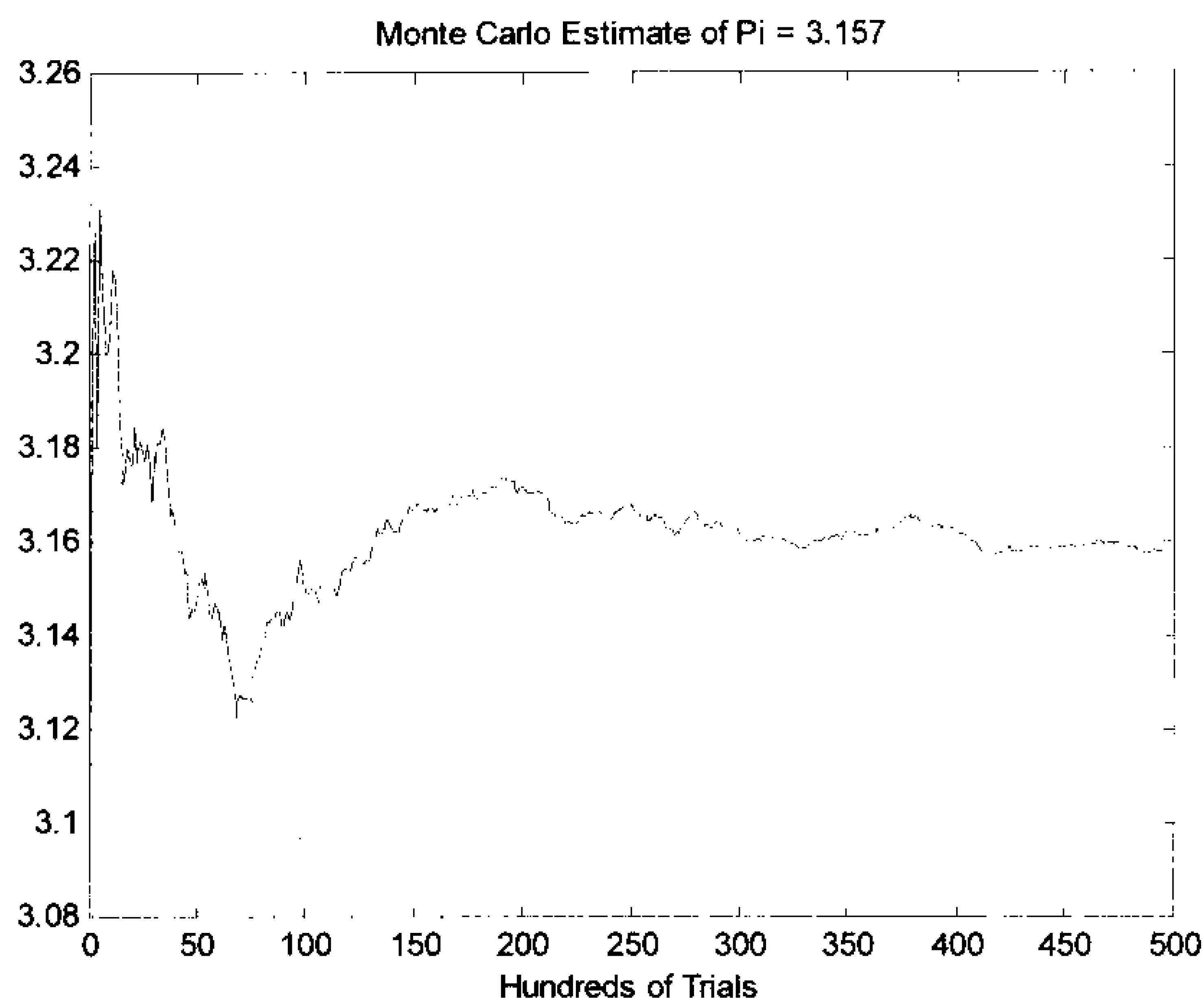


图 2-9 Monte Carlo 评估

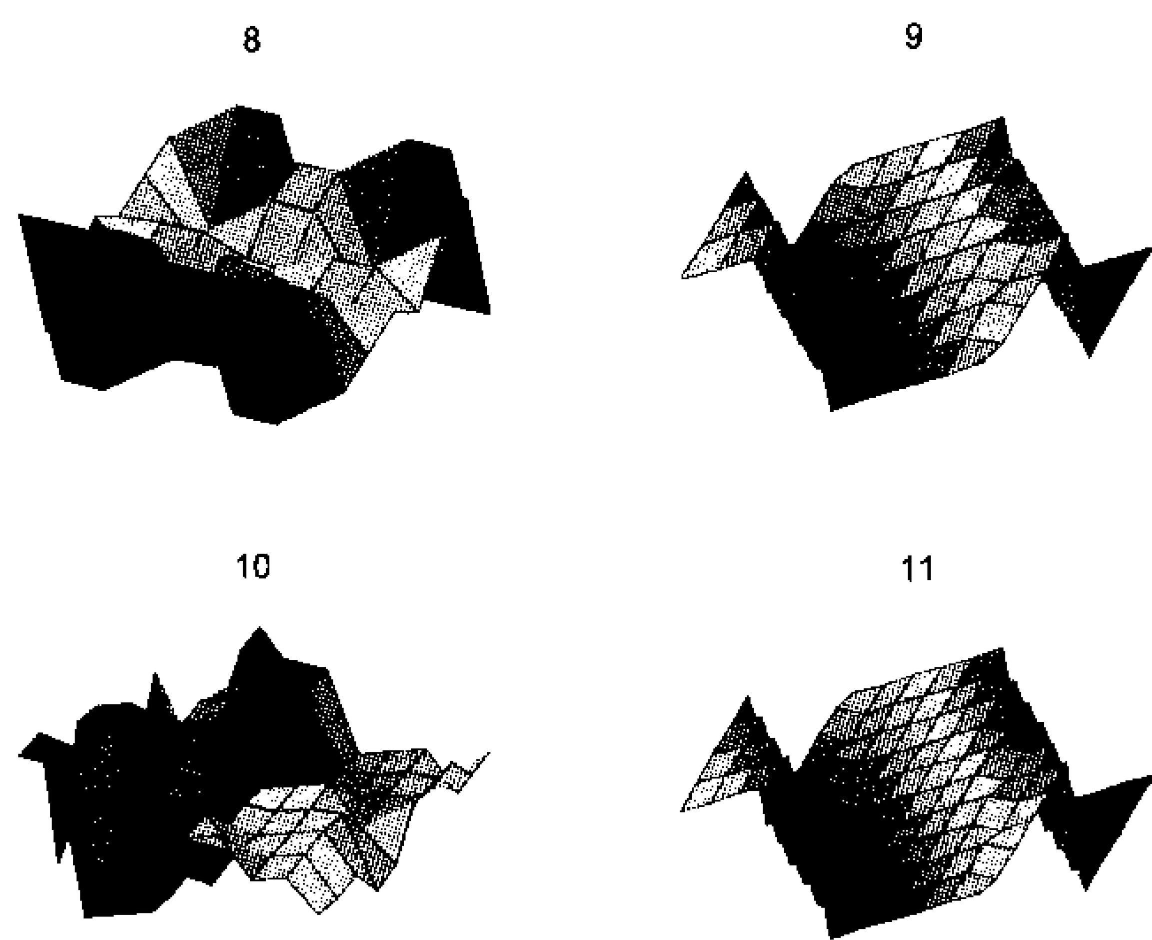


图 2-10 运用 magic 函数产生的三维图形

2.4.4 稀疏矩阵及其应用

稀疏矩阵是一种特殊的矩阵形式，在计算机系统中，内存空间只分配给非零元素。为了能有效地提高计算机的存储效率以及节省计算时间和存储空间，所以产生了稀疏矩阵的理论和方法。MATLAB 中有 4 个最基本稀疏矩阵，它们分别是单位矩阵、随机矩阵、对称随机矩阵和对角矩阵。在 MATLAB 中，提供了一系列的特殊函数来进行稀疏矩阵的运算，如表 2-7 所示。

表 2-7 常用的稀疏矩阵函数

函 数 名	功 能 说 明
sparse	生成一个稀疏矩阵
spdiags	以对角带生成稀疏矩阵
sprandsym	生成稀疏的对称矩阵
sprandn	生成一个稀疏的正态分布随机矩阵
sprand	生成一个稀疏的均匀分布随机矩阵
speye	生成稀疏的单位矩阵
full	把稀疏矩阵转变成满矩阵
find	求非零元素在稀疏矩阵中的索引
spconvert	导入外部数据生成稀疏矩阵
spalloc	给稀疏矩阵分配内存空间
nnz	求稀疏矩阵的非零元素个数
nonzeros	返回一个包含所有非零元素的列向量
spy	查看稀疏矩阵中非零元素分布情况

例 2.52 生成一个 0-1 分布的随机稀疏矩阵。

```
>> a=[1 6 8 4 9 5 7 2];
>> x=diag(a)
x =   1    0    0    0    0    0    0    0
```

```
     0     6     0     0     0     0     0     0
     0     0     8     0     0     0     0     0
     0     0     0     4     0     0     0     0
     0     0     0     0     9     0     0     0
     0     0     0     0     0     5     0     0
     0     0     0     0     0     0     7     0
     0     0     0     0     0     0     0     2
>> y=sprand(x)    %生成一个稀疏的均匀分布的随机矩阵
y =
   (1,1)        0.7922
   (2,2)        0.9595
   (3,3)        0.6557
   (4,4)        0.0357
   (5,5)        0.8491
   (6,6)        0.9340
   (7,7)        0.6787
   (8,8)        0.7577
>> spy(y,'d')
```

然后用函数 spy 查看非零元素的分布情况，如图 2-11 所示。

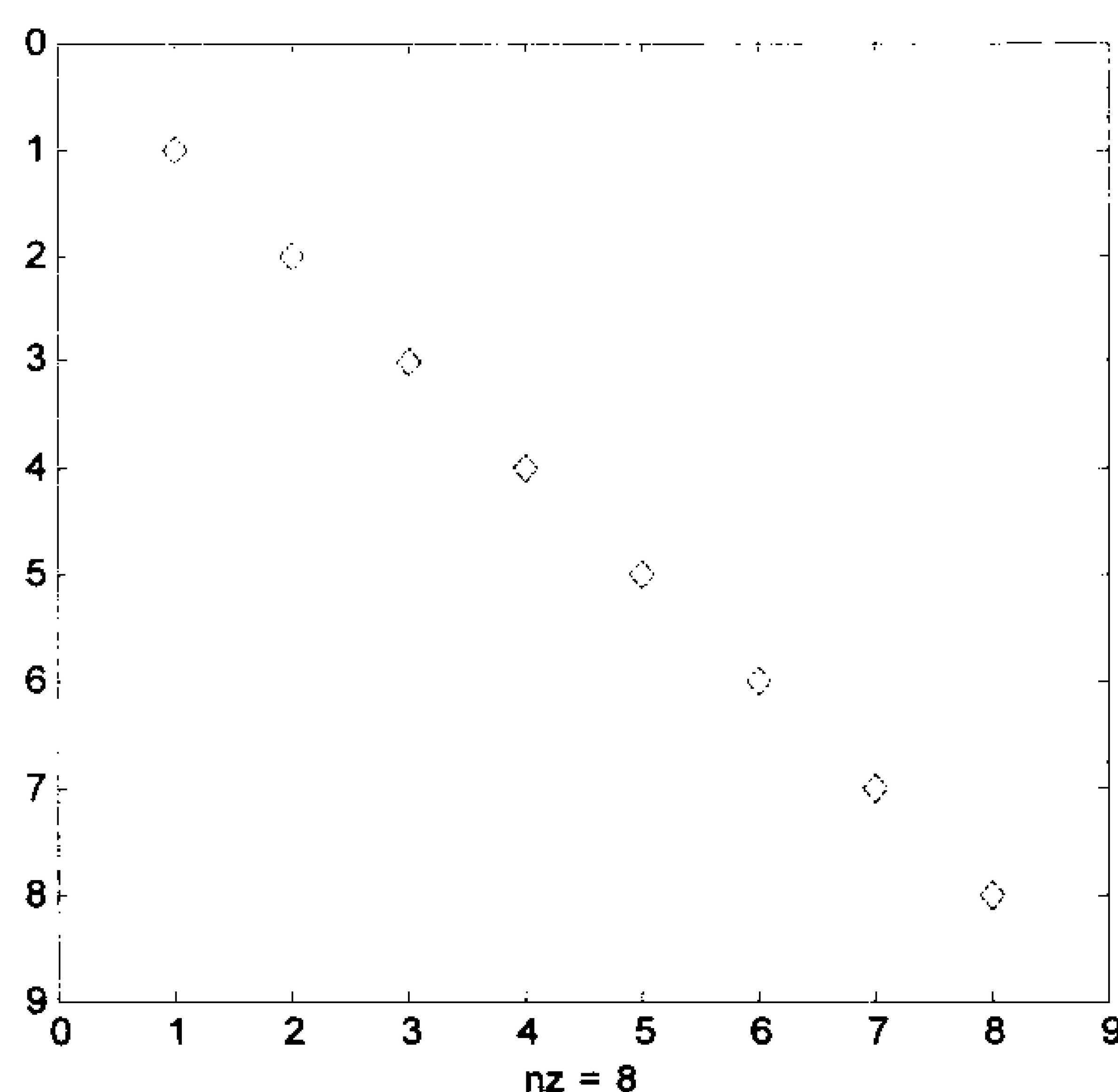


图 2-11　稀疏矩阵分布图

例 2.53　稀疏矩阵的存储。

因为稀疏矩阵的存储是按列存储的，而系统访问矩阵的时候一般是按照行读取效率更高。试比较添加矩阵的行和列所用的时间。

```
>> S = sparse(10000,10000,1);
tic;                              %开始计时
for n = 1:1000
   A = S(100,:) + S(200,:);       %添加行
end;
toc                               %计时结束
Elapsed time is 1.180907 seconds.
>> S = sparse(10000,10000,1);
tic;
for n = 1:1000
```

```
    B = S(:,100) + S(:,200);      %添加列
end;
toc
Elapsed time is 1.095647 seconds.
>> S = sparse(10000,10000,1);
tic;
for n = 1:1000
    A = S(100,:)' + S(200,:)';
    A = A';    %先添加列的转置，然后再转置，因为在矩阵的转置操作时间可以忽略不计
end;
toc
Elapsed time is 0.745029 seconds.
```

从例 2.53 中可以很明显地看出，在编程中若是能够有效利用稀疏矩阵的存储特点将会大大提高运算效率。

2.4.5 矩阵的基本数值运算及其应用

矩阵的基本运算主要包括矩阵和常数的运算、矩阵与矩阵的运算、矩阵的幂运算、指数运算、对数运算、开方运算、矩阵的逆运算，以及矩阵的行列式运算。这里将详细阐述矩阵运算的数值运算以及其与数组运算的异同。

1）矩阵的加法（减法）运算

矩阵的加法（减法）运算包括矩阵与常数之间和矩阵与矩阵之间的加法（减法）运算。前者是指矩阵中各元素与常数之间的运算，后者是指矩阵中各元素之间的加法（减法）运算，需注意的是，在进行矩阵的加法（减法）运算时它们的维数必须相同。此时，矩阵运算和数组运算的运算规则是完全一样的，运算符也是相同的。

例 2.54 矩阵的加法（减法）运算。

```
>> A=rand(3);
>> B=A+10
B =
    10.5290    10.7964    10.0764
    10.5985    10.8226    10.4775
    10.0550    10.2560    10.9543
>> C=magic(3);
>> D=B-C
D =
     2.5290     9.7964     4.0764
     7.5985     5.8226     3.4775
     6.0550     1.2560     8.9543
```

2）矩阵的乘法运算

矩阵的乘法运算不同于数组的乘法运算，它们的运算符号也不相同，前者为“*”，而后者为“.*”。两矩阵在做乘法运算时内维必须一致，除非其中一个为标量；而数组的乘法运算要求两个数组必须大小相同，除非其中一个是标量。我们通过例 2.55 来体会一下二者乘法运算的区别。

例 2.55

```
>> A = pascal(3);
```

```
>>B = magic(3);
>>m = 3; n = 3;
>>for i = 1:m                    %矩阵的乘法运算的本质
      for j = 1:n
          C(i,j) = A(i,:)*B(:,j);
      end
end
>> C
C =
     15     15     15
     26     38     26
     41     70     39
>> X=A*B                          %矩阵的乘法运算
X =
     15     15     15
     26     38     26
     41     70     39
>> Y=B*A
Y =
     15     28     47
     15     34     60
     15     28     43
> Z1=B.*A                              %数组的乘法运算
Z1 =
      8      1      6
      3     10     21
      4     27     12
>> Z2=A.*B        %X≠Y，而 Z1=Z2，数组运算可以交换位置，而矩阵运算不能，即 AB≠BA
Z2 =
      8      1      6
      3     10     21
      4     27     12
```

显然，矩阵的乘法运算为矩阵$\boldsymbol{C}$中的（i，j）元素是对应的矩阵$\boldsymbol{A}$中第i行和矩阵$\boldsymbol{B}$中第j列的乘积累加求和，而数组的乘法运算只是简单的元素和元素之间进行乘法运算。

3）矩阵的除法运算

在 MATLAB 中，矩阵的除法运算有两种：左除和右除，运算符分别为“\”和“/”，二者的概念完全不同。矩阵的除法运算一般可以用来求解方程组的解，但应注意矩阵的左除用来求解线性方程组 A*X=b，其中 X=A\\b；矩阵的右除用来求解线性方程组 X*A=b，其中 X=A/b。

而数组的除法运算符为“.\”和“./”，分别用来运算数组的左除和右除，数组的除法运算是两个数组相应的元素进行相除，且两个数组的大小必须一致，除非其中有一个是标量。

例 2.56 求矩阵 A=magic(3)和 B=[1 3 4;2 1 2;2 1 1]相除并比较其数组除法的不同。

```
>> A=magic(3);
>> B=[1 3 4;2 1 2;2 1 1];
>> C1=A\B       %矩阵的左除，等价于 inv(A)*B
C1 =
     -0.0139      0.3611      0.3639
```

```
        0.1944   -0.0556   -0.0944
        0.1528    0.0278    0.1972
>> C2=A/B        %矩阵的右除，等价于 A*inv(B)
C2 =
       -1.2000    6.2000   -1.6000
        1.4000    0.6000    0.2000
        2.8000   -9.8000   10.4000
>> D1=A.\B       %数组的左除
D1 =
        0.1250    3.0000    0.6667
        0.6667    0.2000    0.2857
        0.5000    0.1111    0.5000
>> D2=A./B       %数组的右除
D2 =
        8.0000    0.3333    1.5000
        1.5000    5.0000    3.5000
        2.0000    9.0000    2.0000
```

从例 2.56 中的结果可以看出矩阵的除法运算和数组的除法运算完全不同。

4）矩阵的幂运算

矩阵的幂运算表达式为“A^B”，其中 A 可以是矩阵或者标量，且 B 不能为矩阵；而数组的幂运算表达式为“A.^B”，其中 A、B 均可以为矩阵或标量，且当 B 为矩阵时，A 与 B 的大小必须一致。

例 2.57

```
>> A=rand(3);
>> B=magic(3);
>> A^B                          %矩阵幂运算时，B 为矩阵系统给出出错信息
??? Error using ==> mpower
At least one operand must be scalar.
>> A^-2                         %矩阵的幂运算
ans =
      -11.9042   16.5775  -10.5979
        0.1078   -5.6185    6.0542
       13.1903  -13.5239    7.0507
>> A.^B                         %B 为矩阵时的数组幂运算
ans =
        0.1551    0.0357    0.0978
        0.8833    0.4414    0.1434
        0.1849    0.5409    0.5522
>> A.^-2                        %B 为标量时的数组幂运算，同样数组的幂运算和矩阵的幂
运算是不同的
ans =
        1.5934  784.1145    2.1707
        1.0862    1.3869    1.7416
        2.3256    1.1463    1.8108
```

5）矩阵的指数运算、对数运算和开方运算

矩阵的指数运算、对数运算和开方运算与前面阐述的数组的指数运算、对数运算和开方运算是不同的，它不是对矩阵中的单个元素的运算，而是对矩阵整体的运算。矩阵的这 3 种运算

分别通过 MATLAB 提供的函数来实现，这与数组的 3 种运算函数 exp、log、sqrt 也是不同的。

例 2.58

```
>> a=[1 2 3 ];
>> A=Vander(a)          %创建范德蒙矩阵 A
A =
     1     1     1
     4     2     1
     9     3     1
>> B=sqrtm(A)           %矩阵的开方运算
B =
   0.8943 + 0.7487i   0.3536 + 0.0000i   0.2912 - 0.2496i
   1.3518 - 0.2496i   1.0607 - 0.0000i   0.3744 + 0.0832i
   2.6828 - 1.9965i   1.0607 - 0.0000i   0.8735 + 0.6655i
>> C=sqrt(A)            %数组的开方运算
C =
    1.0000    1.0000    1.0000
    2.0000    1.4142    1.0000
    3.0000    1.7321    1.0000
```

6）矩阵的逆运算

矩阵的逆运算即求矩阵的逆矩阵，它是矩阵运算中非常重要的运算之一。在线性代数里面求解逆矩阵是一件非常复杂的事情，在 MATLAB 中，提供的函数 inv 能够简便地解决问题。

例 2.59 求矩阵 $\boldsymbol{A}$ 的逆矩阵，其中 $\boldsymbol{A}$ =[4 8 9;7 6 3;1 5 2]。

```
>> A=[4 8 9;7 6 3;1 5 2];
>> B=inv(A)
B =
   -0.0186    0.1801   -0.1863
   -0.0683   -0.0062    0.3168
    0.1801   -0.0745   -0.1988
```

7）矩阵的行列式运算

在 MATLAB 中，直接使用函数 det 可求得矩阵的行列式大小。

例 2.60 求矩阵行列值。

```
A=[4 8 9;7 6 3;1 5 2]
>> C=det(A)
C =
   161
```

8）矩阵基本运算应用举例

例 2.61 计算 $x(t)=e^{tA}x(0)$ 在区间[0,1]的值。其中，初始值 $x(0)=$ [1;1;1]；系数矩阵 A=[0 -6 -1;6 2 -16;-5 20 -10]。

```
>> A=[0 -6 -1;6 2 -16;-5 20 -10];
>> x0=[1;1;1];
>> X = [];
for t = 0:.01:1
    X = [X expm(t*A)*x0];          %利用矩阵的指数运算函数计算 X 的值
End
>> plot3(X(1,:),X(2,:),X(3,:),'x')      %把得到的 X 值绘制成图，如图 2-12 所示
grid on
```

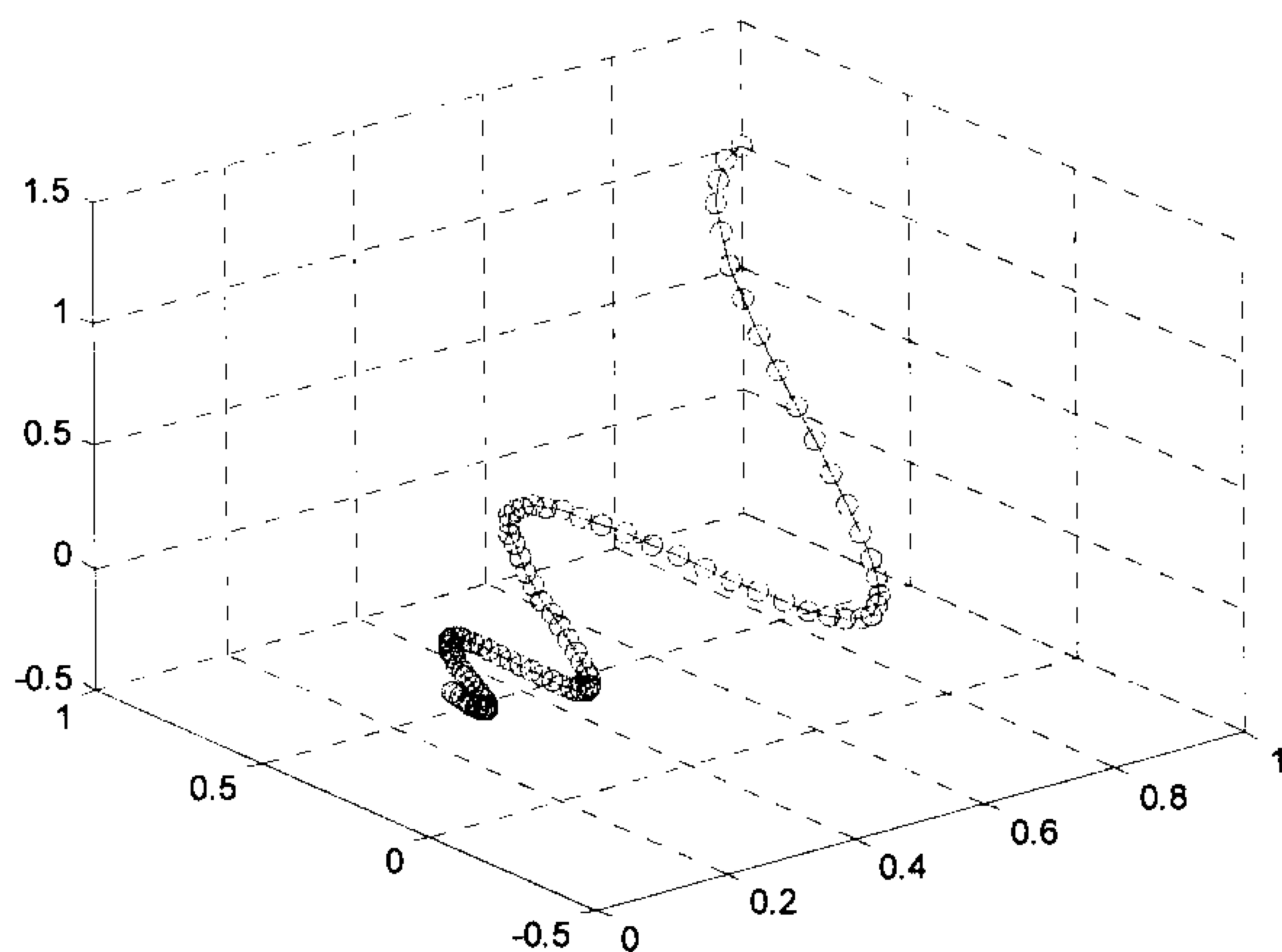


图 2-12 螺旋状图

例 2.62 利用矩阵除法运算求解下面的线性方程组。

$$\begin{cases} 2x_1+5x_2+4x_3=7 \\ -x_1+3x_2=10 \\ x_1-2x_2+6x_3=3 \end{cases}$$

首先把线性方程组变换成矩阵除法运算形式 ***A*** **X*=***B*** 有：***A*** =[2 5 4;-1 3 0;1 -2 6]，***B*** =[7;10;3]。那么可以通过矩阵的左除（\）运算 *X*=***A*** \ ***B*** 来求解方程组 ***A*** **X*=***B*** 。

```
>> A=[2 5 4;-1 3 0;1 -2 6];
>> B=[7;10;3];
>> X=A\B
X =
   -4.6774
    1.7742
    1.8710
```

所以，方程组的解为 *X*=[-4.6774;1.7742;1.8710]。

2.4.6 矩阵的特征参数运算及其应用

特征值和特征向量的求解和运算问题是线性代数中一个重要的课题。它们在工程应用和科学实践中应用非常广泛，下面将介绍常见的矩阵特征参数的运算及其应用。

1）特征值运算

在线性代数里面，求矩阵的特征值也是一件比较麻烦的事情。在 MATLAB 中，矩阵的特征值可以通过函数 eig 和 eigs 来得到。

例 2.63 求魔方矩阵 ***X*** 的特征向量和特征值。

```
>> X=magic(5);
>> E=eig(X)
E =
   65.0000
  -21.2768
  -13.1263
```

```
    21.2768
    13.1263
>> [V,D]=eig(X)
V =
   -0.4472    0.0976   -0.6330    0.6780   -0.2619
   -0.4472    0.3525    0.5895    0.3223   -0.1732
   -0.4472    0.5501   -0.3915   -0.5501    0.3915
   -0.4472   -0.3223    0.1732   -0.3525   -0.5895
   -0.4472   -0.6780    0.2619   -0.0976    0.6330
D =
   65.0000         0         0         0         0
         0  -21.2768         0         0         0
         0         0  -13.1263         0         0
         0         0         0   21.2768         0
         0         0         0         0   13.1263
```

其中，V 和 D 分别为矩阵 X 的特征向量和特征值组成的矩阵。

2）秩运算

矩阵的秩在线性代数中运用也十分广泛，MATLAB 提供求矩阵的秩的函数为 rank。

例 2.64 求随机矩阵 A 的秩。

```
>> A=rand(5)
A =
    0.9575    0.4854    0.7922    0.9340    0.6555
    0.9649    0.8003    0.9595    0.6787    0.1712
    0.1576    0.1419    0.6557    0.7577    0.7060
    0.9706    0.4218    0.0357    0.7431    0.0318
    0.9572    0.9157    0.8491    0.3922    0.2769
>> r=rank(A)
r =
     5
```

3）逆运算及伪逆运算

矩阵的逆，有些资料和书籍上也称为矩阵的迹，指主对角线上所有元素的和。在 MATLAB 中，提供函数 trace 来进行逆运算，伪逆运算的函数为 pinv 。

例 2.65

```
>> clear
>> A=[1 6 8;4 0 9;3 7 5];
>> a=trace(A)
a =
     6
>> b=pinv(A)
b =
   -0.3103    0.1281    0.2660
    0.0345   -0.0936    0.1133
    0.1379    0.0542   -0.1182
```

4）条件数运算

矩阵的条件数表示矩阵的病态程度，根据矩阵的条件数，可以判断矩阵是否“病态”。病态矩阵的概念为：在求解 $\boldsymbol{A}\,\boldsymbol{X}=\boldsymbol{b}$ 时，如果 $\boldsymbol{A}$ 的微小变化会引起解的剧烈变化，则矩阵 $\boldsymbol{A}$ 为病态矩阵。矩阵的条件数可以由函数 cond、condest 和 rcond 计算得到，它们的含义分别

为计算矩阵的条件数（默认为2-范式）和1-范式条件数以及逆条件数。

例2.66 分别求矩阵的2-范式条件数、1-范式条件数和逆条件数。

```
>> clear
>> A=[1 6 8;4 0 9;3 7 5]
>> b=cond(A)      %计算矩阵的2-范式条件数
b =
    7.1072
>> c=condest(A)  %计算矩阵的1-范式条件数
c =
    10.9458
>> d=rcond(A)      %计算矩阵的逆范式条件数
d =
    0.0914
```

例2.67 判断方程组$\begin{cases} x_1+\dfrac{1}{3}x_2+\dfrac{1}{5}x_3=\dfrac{4}{3} \\ \dfrac{1}{2}x_1+\dfrac{1}{4}x_2+\dfrac{1}{6}x_3=\dfrac{7}{6} \\ \dfrac{1}{3}x_1+\dfrac{1}{5}x_2+\dfrac{1}{7}x_3=\dfrac{32}{31} \end{cases}$的性态。

```
>> A=[1 1/3 1/5;1/2 1/4 1/6;1/3 1/5 1/7];
>> rcond(A)
ans =
    0.0013
>> b=[4/3;7/6;32/31];
>> X=A\b
X =
    0.0484
   -2.5806
   10.7258
%将方程组中的A和b做细微的改变成A1和b1，然后求解方程组A1*X=b1
>> A1=[1 0.33 0.20;0.5 0.25 0.17;0.33 0.20 0.14]
A1 =
    1.0000    0.3300    0.2000
    0.5000    0.2500    0.1700
    0.3300    0.2000    0.1400
>> b1=[1.33;1.67;1.03];
>> X1=A1\b1
X1 =
      31.5057
     -379.2299
      474.8506
```

从程序中可以看出，如果A和b的微小变化引起了方程组的解的剧烈变化，则该方程组为严重病态方程组。

5）范数运算

数值分析中引入了范数的概念，它可以分为2-范数、1-范数、无穷范数和Frobenius范数等。范数运算，通过这些范数可以看出矩阵或向量的某些特征。MATLAB提供了norm和normest两个函数来计算矩阵或向量的范数，其中函数normest只能计算矩阵的2-范数值。

例 2.68

```
>> clear
>> a=[5 9 0;7 6 -2;8 4 3]
>> b=norm(a,'inf')              %求矩阵 a 的无穷范数
b =
     15
>> b=norm(a,'fro')              %求矩阵 a 的 Frobenius 范数
b =
    16.8523
>> b=norm(a,2)                  %求矩阵 a 的 2-范数
b =
    15.9605
```

6）奇异值运算

在 MATLAB 中，通过函数 svd 和 svds 可以计算得到矩阵的奇异值。

例 2.69

```
>> a=[5 9 0;7 6 -2;8 4 3]
>> b=svd(a)
b =
    15.9605
     4.6758
     2.7202
```

7）正交化运算

在矩阵论中，判断一个矩阵是否是正交矩阵的充分必要条件是该矩阵的列向量都是单位向量，且两两正交。在 MATLAB 中，通过函数 orth 来求得矩阵的正交矩阵。

例 2.70 求矩阵 ***A*** 的正交矩阵，其中 ***A*** =[25 18 9;30 2 15;8 40 6]。

```
>> A=[25 18 9;30 2 15;8 40 6];
>> B=orth(A)
B =
    -0.5874    -0.1954    -0.7854
    -0.4690    -0.7086     0.5271
    -0.6595     0.6780     0.3246
```

同时，验证矩阵 ***B*** 是否为正交矩阵。

```
>> I=B*B'
I =
     1.0000          0     0.0000
          0     1.0000    -0.0000
     0.0000    -0.0000     1.0000
```

2.4.7 矩阵的分解运算及其应用

MATLAB 拥有强大的数学处理能力，主要是因为它提供大量的矩阵运算函数，这些函数能够帮助用户非常轻松地解决数学计算中那些求解过程复杂的难题。这里将要介绍一些在数值分析中占据重要位置的分解运算。矩阵的分解运算是指将给定的矩阵分解成特殊矩阵的乘积的过程。一般的矩阵分解运算主要有：三角（LU）分解、正交（QR）分解、Chollesky（CHOL）分解、特征值（EIG）分解和奇异值（SVD）分解。下面将一一阐述，并举例说

明分解运算的应用。

1）三角（LU）分解

三角分解是矩阵分解中最常见的分解方法，是方程求解方法中高斯削元法的基础，在线性方程的解法中应用非常广泛。在数值分析中，非奇异矩阵 $\boldsymbol{A}$ （$n\times n$），如果其顺序主子式均不为零，则存在唯一的单位下三角矩阵 $\boldsymbol{L}$ 和上三角矩阵 $\boldsymbol{U}$ ，使得 $\boldsymbol{A}=\boldsymbol{L}*\boldsymbol{U}$ 。在 MATLAB 中，提供函数 lu 来进行三角分解。

格式如下：

[L,U]=lu(X)　　　%X=L*U

[L,U,P]=lu(X)　　%返回一个置换矩阵 P，且满足关系 L*U=P*X

例 2.71　求矩阵 $\boldsymbol{X}$ 三角分解后的矩阵，其中 $\boldsymbol{X}$ =[1 5 8;12 9 3;7 6 10]。

```
>> X=[1 5 8;12 9 3;7 6 10];
>> [L,U,P]=lu(X)
L =                                          %求得的下三角矩阵 L
    1.0000         0         0
    0.0833    1.0000         0
    0.5833    0.1765    1.0000
U =                                          %求得的上三角矩阵 U
   12.0000    9.0000    3.0000
         0    4.2500    7.7500
         0         0    6.8824
P =                                          %求得的置换矩阵 P
     0     1     0
     1     0     0
     0     0     1
```

2）正交（QR）分解

在数值分析中，为了求解矩阵的特征值，引入了正交（QR）分解方法。对于非奇异矩阵 $\boldsymbol{A}$ （$n\times n$），则存在正交矩阵 $\boldsymbol{Q}$ 和上三角矩阵 $\boldsymbol{R}$ ，使得 $\boldsymbol{A}=\boldsymbol{Q}*\boldsymbol{R}$ ，QR 分解是唯一的。MATLAB 中提供的函数为 qr 。

例 2.72　求矩阵 $\boldsymbol{A}$ 的正交分解，其中 $\boldsymbol{A}$ =[1 5 8;12 9 3;7 6 10]。

```
>> A=[1 5 8;12 9 3;7 6 10];
>> [Q,R]=qr(A)
Q =
   -0.0718    0.9858   -0.1516
   -0.8615   -0.1379   -0.4886
   -0.5026    0.0956    0.8592
R =
  -13.9284  -11.1284   -8.1847
         0    4.2614    8.4285
         0         0    5.9136
```

3）Chollesky（CHOL）分解

矩阵 $\boldsymbol{A}$ （$n\times n$）为对称正定时，则存在唯一的对角元素为正的上三角矩阵 $\boldsymbol{R}$ ，使得 $\boldsymbol{R}*\boldsymbol{R}'=\boldsymbol{A}$ ，这种分解就称为 Chollesky（CHOL）分解。在 MATLAB 中，提供的函数为 chol 。

> **注意**　当矩阵 A 为非正定的时候，系统将会出现出错信息。

例 2.73 求矩阵的 Chollesky 分解。

```
>> A=[4 5 9;1 9 3;7 6 2];
>> R=chol(A)
??? Error using ➔ chol
Matrix must be positive definite.  %此时 A 为非正定矩阵，通过求矩阵的特征值就可以知道此矩
阵非正定
>> A=[3 -1 1;-1 5 2;1 2 4];
>> R=chol(A)
R =
     1.7321    -0.5774     0.5774
          0     2.1602     1.0801
          0          0     1.5811
```

4）特征值（EIG）分解

特征值分解利用的依然是 eig 函数，只不过在做矩阵分解时，需要形式上做相应的变化。

格式如下：

[V,D]＝eig（A） %V，D 分别是 A 的特征向量作为列向量组成的矩阵和特征值作为主对角线元素构成的矩阵，且满足关系 A*V=V*D

[V,D]＝eig（A,B） % V，D 分别是 A,B 的广义特征向量矩阵和特征值矩阵，且满足关系 A*V=B*V*D

例 2.74 求矩阵的特征值分解。

```
>> clear
>>A=[4 5 9;1 9 3;7 6 2];
>> B=magic(3);
>> [V,D]=eig(A,B)
V =
    0.5974 + 0.2949i    0.5974 - 0.2949i    1.0000
   -0.0516 - 0.4658i   -0.0516 + 0.4658i    0.8670
   -0.6591 + 0.3409i   -0.6591 - 0.3409i    0.2914
D =
    0.2867 + 1.0211i                   0                   0
                   0    0.2867 - 1.0211i                   0
                   0                   0              1.0322
>> Z=A*V-B*V*D
Z =
   1.0e-014 *
   0.0444 + 0.1998i    0.0444 - 0.1998i    0.3553
  -0.0444 + 0.6661i   -0.0444 - 0.6661i   -0.5329
   0.0444 + 0.4663i    0.0444 - 0.4663i    0.3553
```

5）奇异值（SVD）分解

奇异值分解是线性代数中一种重要的矩阵分解，在信号处理、统计学等领域有重要应用。在 MATLAB 中，矩阵奇异值分解通过函数 svd 来实现。

格式如下：

[U,S,V]=svd（A）%得到一个和 A 的维数相同的正交矩阵 S，还有两个正定矩阵 U 和 V

[U,S,V]=svd（A,0）%对矩阵进行最佳的奇异值分解

例 2.75 求矩阵的奇异值分解。

```
>> A=[4 5 9;1 9 3;7 6 2];
>> [U,S,V]=svd(A)
U =
   -0.6478   -0.7593    0.0620
   -0.5411    0.5159    0.6642
   -0.5363    0.3967   -0.7450
S =
   15.7607         0         0
         0    5.5371         0
         0         0    4.7898
V =
   -0.4369    0.0462   -0.8983
   -0.7186    0.5827    0.3795
   -0.5410   -0.8114    0.2214
```

6）矩阵分解运算应用举例

在 MATLAB 中，线性方程组的求解是通过 3 个基本的矩阵分解运算进行的：LU 分解、QR 分解和 Chollesky 分解，而且，这 3 个运算都是运用三角矩阵，即对角线上方或下方的元素均为零。下面以 LU 分解为例介绍在求解线性方程组中的应用。

正如前面所述，LU 分解可以将矩阵 $\boldsymbol{A}$ 分解成下三角矩阵 $\boldsymbol{L}$ 和上三角矩阵 $\boldsymbol{U}$ 的乘积，即 $\boldsymbol{A}=\boldsymbol{L}\,\boldsymbol{U}$ 。那么在线性系统 $\boldsymbol{A}*x=\boldsymbol{b}$ 中可以通过 LU 分解来快速求解。

因为 $\boldsymbol{A}*x=\boldsymbol{b} \Rightarrow \boldsymbol{LU}*x=\boldsymbol{b}$ ，所以，$x=\boldsymbol{U}\backslash(\boldsymbol{L}\backslash\boldsymbol{b})$。

例 2.76 用 LU 分解求解线性方程组 $\begin{cases} 2x_1+3x_2+4x_3=1 \\ x_1+x_2+9x_3=-7 \\ x_1+2x_2-6x_3=9 \end{cases}$ 。

```
>> A=[2 3 4;1 1 9;1 2 -6];
>> b=[1;-7;9]
b =
     1
    -7
     9
>> [L,U]=lu(A)                    %用 lu()函数分别求得下三角矩阵和上三角矩阵
L =
    1.0000         0         0
    0.5000    1.0000         0
    0.5000   -1.0000    1.0000
U =
    2.0000    3.0000    4.0000
         0   -0.5000    7.0000
         0         0   -1.0000
>> x=U\(L\b)                      %直接用公式求得线性方程组的解
x =
     1
     1
    -1
```

而且，在 MATLAB 中，行列式的运算和矩阵求逆运算都是建立在 LU 分解之上的，即分别有 det(A) = det(L)*det(U)和 inv(A) = inv(U)*inv(L)。

第3章 MATLAB常用运算

数学运算中除了前面的基本数值计算以外，还支持其他许多的科学运算方式，为多领域提供功能强大的函数支持，本章和后面一章将进行介绍。本章着重讲述关系和逻辑运算、多项式运算以及在数学、物理、应用工程和科学计算等方面应用广泛的符号运算。

MATLAB 中的符号运算是 MATLAB 处理数字的自然扩展。MATLAB 中的数组和数值变量用于数值的存储和各种数值计算。MATLAB 的符号数学工具箱提供的工具能够很好地解决符号表达式的复合、简化、微分、积分和求解符号方程以及求解逆、行列式、正则形式的精确结果等问题，并且能求出符号矩阵的特征值而无数值计算引入的误差。工具箱还支持可变精度运算，即支持符号计算并能以指定的精度返回结果。

关系和逻辑运算也是 MATLAB 常用运算中必不可少的一部分，在 MATLAB 程序设计和模糊逻辑推理中常常用到它们。

多项式运算在求解数学问题和工程计算中比较常见，包含多项式求值、多项式求根、多项式部分分式展开、多项式乘除法运算、多项式微积分，以及后面章节中将要详细阐述的多项式数据拟合和多项式插值。

本章首先介绍了 MATLAB 的符号表达式及其基本运算，其次阐述了符号精度的控制、符号矩阵和代数运算、符号微积分、积分变换和方程求解等方法，使读者可以掌握符号的高级运算，再次介绍了 MAPLE 函数的访问，最后阐述了 MATLAB 的关系和逻辑运算、多项式运算及其应用。

本章主要内容：

- 符号运算
- 关系和逻辑运算
- 多项式运算

3.1 符号运算

MATLAB 不仅拥有强大的数值运算能力，还拥有强大的符号运算能力。MATLAB 提供的符号数学工具箱可以完成几乎所有的符号运算功能，并且它可以广泛应用于数学、物理、工程力学等各个学科的科研和工程中。而且，MATLAB 还提供了与 Maple 语言良好的接口，使得 MATLAB 的符号运算更加强大。这一节中将详细阐述符号表达式及其基本运算、符号表达式操作与转换、符号精度的控制、符号矩阵及其运算、符号微积分和积分变换、符号方程求解、符号作图以及 Maple 函数。

3.1.1 符号表达式

符号表达式是代表数字、函数、算子和变量的MATLAB字符串，或字符串数组。不要求变量有预先确定的值，符号方程式是含有等号的符号表达式。符号算术是使用已知的规则和给定符号恒等式求解这些符号方程的实践，它与代数和微积分所学到的求解方法完全一样。符号矩阵是数组，其元素是符号表达式。

MATLAB在内部把符号表达式表示成字符串，以与数字变量或运算相区别；否则，这些符号表达式几乎完全像基本的 MATLAB 命令。表 3-1 中有几个符号表达式例子以及MATLAB等效表达式。

表 3-1 符号表达式及 MATLAB 等效表达式

符号表达式	MATLAB 表达式	符号表达式	MATLAB 表达式
$\frac{1}{2x^n}$	'1/(2*x^n)'	$M=\begin{bmatrix} a & b \\ c & d \end{bmatrix}$	M=sym('[a,b;c,d]')
$\frac{1}{y=\sqrt{2x}}$	y='1/sqrt(2*x)'	$\int_a^b \frac{x^3}{\sqrt{1-x}}dx$	f=int('x^3/sqrt(1-x)','a','b')
$\cos(x^2)-\sin(2x)$	'cos(x^2)-sin(2*x)'		

1. 符号表达式创建

1）创建符号常量

符号常量是不含有变量的符号表达式，符号常量常常与整数很难区分，如例 3.1

一般可以通过命令 sym 来创建。

例 3.1 创建符号常量。

```
>> a=sym('cos(2*x)')      %创建符号常量
a =
cos(2*x)
>> f=simplify((3*4-2)/5+1)      %    reduce a symbolic constant to its simplest form
f =
....3
>> isstr(f)      % is f a string? (1=yes,0=no)
ans=
....0
```

在例 3.1 中，f 代表数字型符号常数 3，但不是字符串型。

2）创建符号变量和符号表达式

在 MATLAB 中，符号变量和符号表达式可以通过函数 sym 和 syms 来创建。

（1）Sym 函数的使用。格式如下：

S = sym(A)

x = sym('x')

x = sym('x','real')

x = sym('x','unreal')

S = sym(A,flag)

Sym 函数可以生成单个的符号变量或表达式，当输入变量为字符串的时候，输出的是

一个符号变量或者一个符号数字；当输入变量为数字向量或者矩阵的时候，输出的就是给出数字值的符号表征。如例 3.2。

例 3.2

```
>> sym1=sym('China')
sym1 =
China
>> sym2=sym('1/10')      %创建符号变量
sym2 =
1/10
>> f=sym('a*x^3+b*x^2+c*x+d')      %创建符号表达式
f =
a*x^3+b*x^2+c*x+d
```

函数 sym 中参数用来设置符号变量或表达式的数学特性，如 x = sym('x','real')表示 x 为实型，同时 x = sym('x','unreal')中的 x 为非实型，S = sym(A,flag)表示当 A 为'r'，'d'，'e'或者'f'格式其中之一时，sym 函数把它从数字向量或者矩阵形式转变成符号对象形式。

（2）Syms 函数的使用。当需要一次生成多个符号对象的时候，常常使用 Syms 函数。格式如下：

```
syms arg1 arg2 ...
syms arg1 arg2 ... real
syms arg1 arg2 ... unreal
syms arg1 arg2 ... positive
```

例 3.3

```
>>
>> whos
  Name      Size           Bytes    Class     Attributes
  Beta      1x1            132      sym
  x         1x1            126      sym
>> clear
>>
>> whos
  Name      Size           Bytes    Class     Attributes
  Beta      1x1            132      sym
  X         1x1            126      sym
```

从例 whos 命令查看的变量信息中可以看出 syms x beta unreal 等价于 x = sym('x','real'); beta = sym('beta','real')。

3）创建符号矩阵

例 3.4　用 sym 函数创建符号矩阵。

```
>> M=[a,b;c,d]      % M is a numeric matrix using value of a through d
??? Undefined function or variable 'a'.
>> M='[a,b;c,d]'      % M is a character string,but not a symbolic matrix
M=
[a,b;c,d]
>> M=sym('[a,b;c,d]')      % M is a symbolic matrix
M=
[ a, b]
[ c, d]
```

M 以 3 种方式定义：数字型（如果 a、b、c、d 已预先确定）、字符串型和符号矩阵型。许多符号函数能够非常巧妙地自动将字符转变为符号表达式。但在某些情况下，尤其是建立符号数组时，必须用函数 sym，特别地将字符串变为符号表达式。隐含形式，例如 diff cos(x)，对于那些不需要参考先前结果的简单任务最有用。但是最简单形式（无引号）要求一个参量，它是一个单字符的字符串、不包含插入的空格。

例 3.5 体会符号矩阵和字符串矩阵的不同。

```
>> clear
>> A=sym('[a,b;c,d]')  %符号矩阵的创建方法
 A =
[ a, b]
[ c, d]
>> B='[a,b;c,d]'         %表达式矩阵的创建方法
B =
[a,b;c,d]
>> C=[a,b;c,d]           %数组矩阵的创建方法，因为数值变量并未先赋值，故出现错误信息提示
??? Undefined function or variable 'a'.
```

2. 符号表达式函数操作

MATLAB 符号函数可让用户用多种方法来操作这些表达式。

例 3.6 符号表达式的函数操作。

```
>> diff('cos(x)')      % differentiate cos(x) with respect to x
ans=
-sin(x)
>> M=sym('[a,b;c,d]')      % create a symbolic matrix M
M=
[ a, b]
[ c, d]
>> det(M)      % find the determinant of the symbolic matrix M
ans=
a*d-b*c
```

注意 上面第一个例子的符号表达式是用单引号以隐含方式定义的。它告诉 MATLAB ‘cos(x)’是一个字符串并说明 diff(‘cosx’)是一个符号表达式而不是数字表达式；然而在第二个例子中，用函数 sym 显式地告诉 MATLAB M=sym(‘[a,b;c,d]’)是一符号表达式。在 MATLAB 可以自己确定变量类型的场合下，通常不要求显式函数 sym。

MATLAB 中函数 function argument 形式是与 function(‘argument’)等价的。其中，function 是一个函数，argument 是一字符串。例如，MATLAB 可以构造 diff cos(x)和 diff(‘cos(x)’)二者都意味 diff (sym’cos(x)’)。显然第一种形式更便于输入。然而，很多时候 sym 是必要的。

例 3.7 函数将字符串变成符号表达式。

```
>> diff x^2+3*x+5      % the argument is equivalent to 'x^2+3*x+5'
ans=
2*x+3
>> diff x^2 + 3*x + 5      % spaces break the argument into separate strings
??? Error using ==> sym.diff at 21
DIFF can be called with at most 3 input arguments
```

对于符号表达式，除去 i 和 j 的小写字母外，其他字母都可以作为变量。在 MATLAB 中，可以利用函数 ***symvar*** 询问 MATLAB 在符号表达式中认为哪一个变量是独立变量。

例 3.8 变量的询问。

```
>> symvar('a*x+y')        % find symbolic variable
ans=
     'a'
     'x'
     'y'
>> symvar('sin(omega)')       %  'omega'   is not a singlee character。
ans=
     'omega'
>> symvar('3*i+4*j')       % i and j are equel to sqrt(-1)
ans=
    Empty cell array: 0-by-1
```

3.1.2 符号表达式的操作与代数运算

在 MATLAB 中，符号表达式的操作和运算主要通过符号函数来实现。所有符号函数（很少特殊例外的情况，讨论于后）作用到符号表达式和符号数组，并返回符号表达式或数组。其结果有时可能看起来像一个数字，但事实上它是一个内部用字符串表示的符号表达式。这一小节将详细阐述符号表达式一些常见的操作和代数运算。

1．符号表达式的操作

1）符号表达式中自由变量的确定

如果不能确定符号表达式中的自由符号变量，一般可以通过 MATLAB 提供的函数 findsym 来确定。

例 3.9 获取符号表达式中的符号变量。

```
>> f=sym('a*x^3+b*x^2+c*x+d')
f=
a*x^3+b*x^2+c*x+d
>> findsym(f)                  %获得符号表达式中所有的符号变量
ans =
a, b, c, d, x
>> syms a x y z t
>> f=sym('cos(2*pi*x)+tan(y)+sin(pi*t)')
f=
cos(2*pi*x)+tan(y)+sin(pi*t)
>> findsym(f,2)               %获得符号表达式中的两个符号变量
ans =
x,y,
```

2）提取分子和分母

如果表达式是一个有理分式（两个多项式之比），或是可以展开为有理分式（包括哪些分母为 1 的分式），可利用函数 numden 来提取分子或分母。例如，给定如下的表达式：

$$m = x^2 \quad f = \frac{ax^2}{b-x} \quad g = \frac{3}{2}x^2 + \frac{2}{3}x - \frac{3}{5} \quad h = \frac{x^2+3}{2x-1} + \frac{3x}{x-1} \quad k = \begin{cases} \frac{3}{2} & \frac{2x+1}{3} \\ \frac{4}{x^2} & 3x+4 \end{cases}$$

在必要时，numden 将表达式合并、有理化并返回所得的分子和分母。

例 3.10 分子和分母的提取。

```
>> g=sym('3/2*x^2+2/3*x-3/5')      % rationalize and extract the parts
g=
3/2*x^2+2/3*x-3/5
>> [n,d]= numden(g)
n=
45*x^2+20*x-18
d=
30
>> h=sym('(x^2+3)/(2*x-1)+3*x/(x-1)')      % the sum of rational polynomials
h=
(x^2+3)/(2*x-1)+3*x/(x-1)
>> [n,d]= numden h)      % rationalize and extract
n=
x^3+5*x^2-3
d=
(2*x-1)*(x-1)
```

在提取各部分之前，这两个表达式 g 和 h 被有理化，并变换成具有分子和分母的一个简单表达式。

例 3.11 分子和分母数组的提取。

```
>> k=sym('[3/2,(2*x+1)/3;4/x^2,3*x+4]')      % try a symbolic array
k=
[          3/2, (2*x+1)/3]
[        4/x^2,        3*x+4]
>> [n,d]=numden(k)
n =
[        3, 2*x+1]
[        4, 3*x+4]
d =
[    2,    3]
[ x^2,     1]
```

这个表达式 k 是符号数组，numden 返回两个新数组 n 和 d，其中 n 是分子数组，d 是分母数组。如果采用 s=numden(f)形式，numden 仅把分子返回到变量 s 中。

3）符号表达式的化简

MATLAB 中的符号表达式有时相当复杂，不便于进行各种操作和书写。为此，MATLAB 提供了几个命令函数用来化简复杂的符号表达式，主要有 pretty、collect、expand、horner、factor、simplify 和 simple 等。

对于同一符号表达式一般有以下 3 种表达形式。

- 多项式形式：$f(x) = x^3 - 6x^2 + 11x - 6$
- 因式形式：$g(x) = (x-1)(x-2)(x-3)$

- 嵌套形式：$h(x)=x(x(x-6)+11)-6$

例 3.12　给定如下 3 个符号表达式。

```
>> f=sym('x^3-6*x^2+11*x-6');
>> g=sym('(x-1)*(x-2)*(x-3)');
>> h=sym('-6+(11+(-6+x)*x)*x');
```

通过使用 pretty 命令，可以得到如下形式的符号表达式（pretty 主要用于将符号表达式以常用方式显示）。

```
>> pretty(f)
          3      2
         x - 6x + 11x - 6
>> pretty(g)
         (x - 1) (x - 2) (x - 3)
>> pretty(h)
         -6 + (11 + (-6 + x) x) x
```

为了使 g 和 h 的表达式以更为简捷的形式表示，可以使用如下命令。

```
>> collect(f)
ans =
x^3-6*x^2+11*x-6
>> collect(g)
ans =
x^3-6*x^2+11*x-6
>> collect(h)
ans =
x^3-6*x^2+11*x-6
```

可见，实际上 f、g、h 都是同一个符号表达式，只是书写格式不同而已。

从上面的例子可以看出，collect 命令主要是将表达式中同类项合并，合并后的多项式以变量幂的次数按大小依次排列。collect 常使用如下两种命令格式。

- collect(S)：对于多项式 S 中的每一个函数，collect(S)按默认变量 x 的次数合并系数。
- collect(S,v)：对指定的变量 v 计算，collect(S)按默认变量 v 的次数合并系数。

该函数功能如表 3-2所示。

表 3-2　collect 函数功能表

f	collect(f)	f	collect(f)
(x-1)*(x-2)*(x-3)	x^3-6*x^2+11*x-6	(1+x)*t + x*t	2*x*t+t
x*(x*(x-6)+11)-6	x^3-6*x^2+11*x-6		

函数 expand 主要用于展开符号表达式中的各项子式，其命令格式如下。

expand(S)：对符号表达式 S 中每个因式的乘积进行展开计算。

该命令通常用于计算多项式函数、三角函数、指数函数与对数函数等表达式的展开式。该函数功能如表 3-3所示。

表 3-3　expand 函数功能表

f	expand(f)	f	expand(f)
a*(x + y)	a*x + a*y	exp(a+b)	exp(a)*exp(b)
(x-1)*(x-2)*(x-3)	x^3-6*x^2+11*x-6	cos(x+y)	cos(x)*cos(y)-sin(x)*sin(y)
x*(x*(x-6)+11)-6	x^3-6*x^2+11*x-6	cos(3*acos(x))	4*x^3-3*x

factor 命令主要用于符号因式分解，其命令格式如下。

factor(X)：参量 X 可以是正整数、符号表达式阵列或符号整数阵列。若 X 为一正整数，则 factor(X)返回 X 的质数分解式。若 X 为多项式或整数矩阵，则 factor(X)分解矩阵的每一元素。若整数阵列中有一元素位数超过 16 位，用户必须用命令 sym 生成该元素。该函数功能如表 3-4所示。

表 3-4 factor 函数功能表

f	factor (*f*)	*f*	factor (*f*)
x ^3-6* *x* ^2+11* *x* -6	6(*x* -1)*(*x*-2)*(*x*-3)	*x* ^6+1	(*x* ^2+1)*(*x*^4 - *x*^2 + 1)
x ^3-6* *x* ^2+11* *x* -5	-5 *x* ^3-6* *x* ^2+11* *x* -5		

simplify 是 MATLAB 化简中使用很多的一个命令，它主要用来在符号表达式中进行等式的恒等替换。该函数功能如表 3-5所示。

表 3-5 simplify 函数功能表

f	simplify(*f*)	*f*	simplify(*f*)
x *(*x**(*x*-6)+11)-6	*x*^3-6**x*^2+11**x*-6	exp(*x*) * exp(*y*)	exp(*x*+*y*)
(1- *x*^2)/(1-*x*)	*x*+1	cos(*x*)^2 + sin(*x*)^2	1
(1/*a*^3+6/*a*^2+12/*a*+8)^(1/3)	((2**a*+1)^3/*a*^3)^(1/3)	syms *x y*	log(*x*)+log(*y*)

其他几个命令也主要用于符号表达式之间的形式转换，读者可自行试验其主要功能。

4）符号表达式的替换

假设有一个以 x 为变量的符号表达式，并希望将变量转换为 y。符号表达式的化简有时候也可以通过替换来实现。MATLAB 提供了 subs 函数，用来在符号表达式中进行变量替换。它们的格式为 subs（f,new,old)，其中 f 是符号表达式，new 和 old 是字符、字符串或其他符号表达式。‘新’ 字符串将代替表达式 f 中各个 ‘旧’ 字符串；还提供函数 subexpr 来替换符号表达式中的子表达式。

例 3.13 subs 函数的变量替换。

```
>> f='a*x^2+b*x+c'      % create a function f(x)
f=
a*x^2+b*x+c
>> subs(f,'s','x')      % substitute ‘s’ for ‘x’ in the expression f
ans=
a*(s)^2+b*(s)+c
>> subs(f,'alpha',' a ')      % substitute ‘alpha’ for ‘a’ in   f
ans=
(alpha)*x^2+b*x+c
```

例 3.14 用 subexpr 函数对子表达式进行替换。

```
>> syms a b c d W
>> [V,D]=eig([a b;c d])
 V =
 [-(1/2*d-1/2*a-1/2*(d^2-2*a*d+a^2+4*c*b)^(1/2))/c,
 -(1/2*d-1/2*a+1/2*(d^2-2*a*d+a^2+4*c*b)^(1/2))/c]
```

```
[1,   1]
D =
[1/2*d+1/2*a+1/2*(d^2-2*a*d+a^2+4*c*b)^(1/2),   0]
[ 0,   1/2*d+1/2*a-1/2*(d^2-2*a*d+a^2+4*c*b)^(1/2)]
>> [R,W]=subexpr([V;D],W)
R =
[ -(1/2*d-1/2*a-1/2*W)/c, -(1/2*d-1/2*a+1/2*W)/c]
[                      1,                      1]
[        1/2*d+1/2*a+1/2*W,                    0]
[                      0,     1/2*d+1/2*a-1/2*W]
 W =
 (d^2-2*a*d+a^2+4*c*b)^(1/2)
```

5）符号函数的求反和复合

MATLAB 具有对符号表达式执行更高级运算的功能。对于函数 $f(x)$，存在另一函数 $g(x)$ 使得函数 $g(f(x))=x$ 成立，则函数 $g(x)$称为函数 $f(x)$的反函数。在 MATLAB 中提供函数 Finverse 来求符号函数的反函数。而用函数 compose 把 $f(x)$和 $g(x)$复合成 $f(g(x))$，以及用函数 symsum 求表达式的符号和。

- 符号表达式函数 $f(x)$的反函数 $g(x)$，满足 $g(f(x))=x$。例如，函数 e^x 的反函数是 $\ln(x)$，因为 $\ln(e^x)=x$。函数 $\sin(x)$的反函数是 $\arcsin(x)$，函数 $\frac{1}{\tan(x)}$ 的反函数是 $\arcsin(\frac{1}{x})$。函数 finverse 返回表达式的反函数，如果解不是唯一就给出警告。

例 3.15 求函数的反函数。

```
>> finverse(sym('1/x'))      % the inverse of 1/x is 1/x since  '1/（1/x)=x'
ans=
1/x
>> finverse(sym('x^2'))      % g(x^2)=x has more than one solution
Warning: finverse(x^2) is not unique.
> In sym.finverse at 48
   In exa3_10 at 3
ans =
x^(1/2)
>> finverse(sym('a*x+b'))      % find the solution to 'g(f(x))=x'
ans=
-(b-x)/a
>> finverse(sym('a*b+c*d-a*z'),sym('a'))      % find the solution to 'g(f(a))=a'
ans=
-(c*d-a)/(b-z)
```

- 在 MATLAB 中用函数 compose 把 $f(x)$和 $g(x)$复合成 $f(g(x))$。

例 3.16 复合下面给定的符号表达式。

$$f=\frac{1}{1+x^2} \qquad g=\sin(x) \qquad h=\frac{1}{1+u^2} \qquad k=\sin(v)$$

```
>> f=sym('1/(1+x^2)');       % create the four expression
>> g=sym('sin(x)');
>> h=sym('1/(1+u^2)');
>> k=sym('sin(v)');
>> compose(f,g)        % find an expression for f(g(x))
ans=
```

```
1/(1+sin(x)^2)
>> compose(g,f)         % find an expression for g(f(x))
ans=
sin(1/(1+x^2))
```

compose 也可用于含有不同独立变量的函数表达式。

```
>> compose(h,k,'u','v')      %   given h(u), k(v), find(k(v))
ans=
1/(1+sin(v)^2)
```

- 用symsun函数求表达式的符号和有4种形式：symsun(f)返回 $\sum_{0}^{x-1} f(x)$；symsum(f, 's') 返回 $\sum_{0}^{s-1} f(s)$；symsun(f，a，b)返回 $\sum_{a}^{b} f(x)$；最普通的形式 symsun(f，'s'，a，b) 返回 $\sum_{a}^{b} f(s)$。

例 3.17 对于 $\sum_{0}^{x-1} x^2$，应返回 $\frac{x^3}{3}-\frac{x^2}{2}+\frac{x}{6}$。

```
>> symsum(sym('x^2'))
ans=
1/3*x^3-1/2*x^2+1/6*x
```

例 3.18 对于 $\sum_{1}^{n} (2n-1)^2$，应返回 $\frac{n(2n-1)(2n+1)}{3}$。

```
>> symsum(sym('(2*n-1)^2'),1,'n')
ans=
11/3*n+8/3-4*(n+1)^2+4/3*(n+1)^3
>> factor(ans)      % change the form ( we will revisit 'factor' later on)
ans=
1/3*n*(2*n-1)*(2*n+1)
```

例 3.19 对于 $\sum_{1}^{\infty} \frac{1}{(2n-1)^2}$ 应返回 $\frac{\pi^2}{8}$。

```
>> symsum(sym('1/(2*n-1)^2'),1,inf)
ans=
1/8*pi^2
```

6）符号对象的数值转换

在 MATLAB 中有 3 种不同的数据类型：数值、符号和字符。每种数据类型都有自己独特的操作函数和指令，为了能够使交换不同的数据类型的数据，MATLAB 提供一些函数来实现这一操作。比如符号表达式变换成数值或反之。有极少数的符号函数可返回数值。然而，某些符号函数能自动地将一个数字变换成它的符号表达式，如果该数字是函数许多参量中的一个。

函数 sym 可获取一个数字参量并将其转换为符号表达式。函数 eval 可将字符串传给 MATLAB 以便计算，所以 eval 是另一个可用于把符号常数变换为数字或计算表达式的函数。

例 3.20 符号常数变成数值。

```
>> phi=sym('(1+sqrt(5))/2')      % the 'golden' ratio
phi=
(1+sqrt(5))/2
```

```
>> eval(phi)     % convert to a numeric value
ans=
    1.6180
```

符号函数 sym2poly 将符号多项式变换成它的 MATLAB 等价系数向量。函数 poly2sym 功能正好相反，并让用户指定用于所得结果表达式中的变量。

例 3.21 符号多项式变成等价系数向量。

```
>> f=sym('2*x^2+x^3-3*x+5')      % f is the symbolic polynomials
f=
2*x^2+x^3-3*x+5
>> n=sym2poly(f)      % extract eht numeric coefficient vector
n=
      1      2      -3      5
>> poly2sym(n)      % recreate the polynomials in x (the default)
ans=
2*x^2+x^3-3*x+5
>> poly2sym(n,' s ')      % recreate the polynomials in s
ans=
s^3+2*s^2-3*s+5
```

2．符号表达式的代数运算

因为 MATLAB 重载技术的应用，使得符号表达式的运算符和函数与数值计算基本上相同。但是符号表达式的代数运算与数值运算相比还是有区别：（1）符号运算的计算结果要比数值运算的结果精确；（2）符号运算可以得到一个完全的封闭解或任意精度的数值解；（3）符号运算占用的内存空间比较大且运算时间比较长。

许多标准的代数运算可以在符号表达式上执行，函数 symadd、symsub、symlnul 和 symdiv 为加、减、乘、除两个表达式，sympow 将一个表达式上升为另一个表达式的幂次。

例 3.22 给定以下两个函数。

$f = 2x^2 + 3x - 5 \qquad g = x^2 - x + 7$

```
>> f=sym('2*x^2+3*x-5')      % define the symbolic expression
f=
2*x^2+3*x-5
>> g=sym('x^2-x+7')
g=
x^2-x+7
>> f+g
ans=
3*x^2+2*x+2
>> f-g
ans=
x^2+4*x-12
>> f*g
ans=
(2*x^2+3*x-5)*(x^2-x+7)
>> f/g
ans=
(2*x^2+3*x-5)/(x^2-x+7)
>> f^3
```

```
ans=
(2*x^2+3*x-5)^3
```

3.1.3 符号精度的控制

不同种类的计算机可能有不同的字长，数值计算的精度在很大程度上受到计算机字长的限制，每次数值操作由于字长的影响都会存在一定的计算误差。在 MATLAB 中，每个算术操作结果的相对精度约为 16 位，但符号计算中的结果是绝对准确的，不包含任何误差。

在 MATLAB 语言中，有 3 种针对不同精度的算法，它们可以视在不同的情况下实现不同的功能：针对浮点运算的数值算法；针对精确运算的符号算法；针对任意精度的符号算法。

1）针对浮点运算的数值算法

在 3 种运算方法中，浮点运算是使用得最多的算法。它的优点就是使用比较简单，但是由于浮点运算在计算机系统中是以多个二进制数表示的，所以，它在 MATLAB 中的表达式或者显示的计算结果都是一个被截断的近似值。

例 3.23 浮点数的精度显示。

```
>> x=1+1/5+1/7+0.02
x =
      1.3629      %得到的结果是取小数点后的 4 位数的值，与真实值存在一点误差
```

2）针对精确运算的符号算法

针对精确运算的符号算法不同于针对浮点运算的数值算法，它能够保证数值的精度。故采用符号算法可以有很高的精度，但是符号运算要占用系统很多的时间和内存资源。

例 3.24 完全精确的符号算法。

```
>> x2=sym(1+1/5+1/7+0.02)
 x2 =
      477/350
```

可以看出采用符号运算结果仍然是分式形式，没有产生任何误差。

3）针对任意精度的符号算法

对于任意精度的符号算法运算，MATLAB 提供的函数主要有：digits 和 vpa。函数 digits 的指令格式如下：

- digits(d)：设置当前的可变算术精度的位数为整数 *d* 位。
- d = digits：返回当前的可变算术精度位数赋值给 *d*。
- digits：显示当前可变算术精度的位数。

还有一个函数，它可以以任何精度实行单个计算，而使全局的 digits 参数不变。即可变精度的算术或函数 vpa，它以默认的精度或任何指定的精度对单个符号表达式进行计算，并以同样的精度来显示结果。

例 3.25 符号精度控制。

```
>>z = 1.0e-16      % z 为一个很小的数
>>x = 1.0e+2      % x 为较大的数
>>digits(14)
>>y1 = vpa(x*z+1)      % 大数 1“吃掉”小数 x*y
>>digits(15)
```

```
>>y2 = vpa(x*z+1)      % 防止“去掉”小数 x*y
z =
   1.0000e-016
x =
   100
y1 =
     1.0000000000000
y2 =
     1.00000000000001
```

例 3.26　符号精度控制。

```
>> pi
ans =
     3.1416
>> format long
>> pi
ans =
   3.141592653589793
>> digits(16)
>> vpa('pi')
ans =
3.141592653589793
>> vpa('pi',200)
ans =
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899
86280348253421170679821480865132823066470938446095505822317253594081284811174502841027019385
2110555964462294895493038 20
>> vpa('pi')
ans =
3.141592653589793
```

3.1.4　符号矩阵及其运算

创建符号矩阵的方法比创建符号表达式、创建符号方程的方法要多一些。总的来说通常有以下几种。

- 用 sym 命令直接创建符号矩阵。
- 用类似创建普通数值矩阵的方法创建符号矩阵。
- 用数值矩阵转换为符号矩阵和以矩阵元素的通式来创建符号矩阵。

符号矩阵可以用有理式和无理式方式来表示。当然，符号矩阵可以进行代数计算。下面将对符号矩阵的创建、表示和代数运算进行论述。

1．符号矩阵

在 MATLAB 中，符号矩阵的元素为符号表达式，同样可用函数 sym 来产生。

例 3.27　直接产生符号矩阵。

```
>> S=sym('[x,y,z;u,v,w;a,b,c]')
S =
[ x, y, z]
[ u, v, w]
```

```
[ a, b, c]
>> H=sym('[cos(t), sin(t);-sin(t),cos(t)]')
H=
[  cos(t),   sin(t)]
[ -sin(t),   cos(t)]
```

函数 sym 也可以把数值矩阵转换成符号矩阵。先建立一个数值矩阵，再通过 sym 命令可直接将数值矩阵转换为符号矩阵。

例 3.28 数值矩阵转换为符号矩阵。

```
>> M=[1.1,1.2,1.3;2.1,2.2,2.3;3.1,3.2,3.3]
M=
     1.1000     1.2000     1.3000
     2.1000     2.2000     2.3000
     3.1000     3.2000     3.3000
>> S=sym(M)
S=
[ 11/10,    6/5, 13/10]
[ 21/10,   11/5, 23/10]
[ 31/10,   16/5, 33/10]
```

如果数值矩阵的元素可以指定为小的整数之比，则函数 sym 将采用有理分式表示。如果元素是无理数，则在符号形式中 sym 将用符号浮点数表示元素。

例 3.29 有理数和无理数的符号表示。

```
>> A=[sin(1) cos(2)]
A =
     0.8415    -0.4161
>> sym(A)
ans =
       [  7579296827247854*2^(-53), -7496634952020485*2^(-54)]
```

用函数 size 可以得到符号矩阵的大小（即行、列数）。函数返回数值或向量，而不是符号表达式。函数 size 的 4 种形式说明如下。

例 3.30 符号矩阵行、列数计算。

```
>> A=[sin(1) cos(2)];
>> s=size(A)
s =
     1     2
>> [s_r,s_c]=size(A)
s_r =
     1
s_c =
     2
>> s_r=size(A,1)
s_r =
     1
>> s_c=size(A,2)
s_c =
     2
```

数值数组用 $N(m,n)$形式来访问单个元素，而在 MATLAB 以前的版本中符号数组元素必须用函数如 sym(S,m,n)来获取。在 MATLAB 中可以直接使用数值数组访问单个元素的方

法访问符号矩阵中的元素。

2．符号矩阵的代数运算

MATLAB 采用了重载技术，用来构成符号计算表达式的运算符和基本函数，无论在形状、名称还是使用方法上都与数值计算中的运算符和基本函数几乎完全相同。

对于基本运算符，如“+”、“-”、“*”、“\”和“^”等都可以直接用于矩阵。还有一些运算符如“.*”、“.\”和“.^”等则分别用于数组的乘、除和求幂。此外，运算符“'”和“.'”分别用于矩阵的共轭转置和非共轭转置。

在三角函数及其相关函数中，除了 atan2 不能作用于符号矩阵外，其他都能直接作用于符号矩阵。

指数函数中符号计算与数值计算完全相同，但对数函数中符号计算只能使用自然对数 log，而不能使用数值计算中的 log2 和 log10。

在 MATLAB 中使用函数 inv 和 det 可计算符号矩阵的逆阵及行列式。

例 3.31 代数矩阵的逆阵及行列式计算。

首先，我们建立一个 3 行 3 列的对称矩阵。

```
>> H=sym(hilb(3))
H=
[   1, 1/2, 1/3]
[ 1/2, 1/3, 1/4]
[ 1/3, 1/4, 1/5]
```

通过命令 det 计算其行列式值。

```
>> det(H)
ans=
1/2160
```

下面使用 inv 命令求得 ***H*** 的逆矩阵。

```
>> J=inv(H)
J=
[    9,  -36,   30]
[  -36,  192, -180]
[   30, -180,  180]
>> det(J)
ans=
2160
```

3.1.5 符号微积分与积分变换

微分和积分是微积分学研究和应用的核心，并广泛地用在许多工程学科中。MATLAB 符号工具能帮助解决许多这类问题。在 MATLAB 中，提供了函数 diff 来进行符号的微分和求导运算，而通过 Jacobian 函数来实现对多元符号函数的求导。而积分变换在工程、应用数学、线性系统等领域都有着重要的作用。这一小节分别讨论符号极限、符号级数、符号的微分和积分运算，以及积分的三大变换。

1．符号极限

求符号表达式的极限是比较常见的操作，MATLAB 符号数学工具箱提供求符号表达式

的极限函数 limit。

格式如下：

- limit(F,x,a)　　%对 x 求趋于 a 的极限，但是当左右极限不相同时，极限不存在
- limit(F,a)　　%用 findsym(F)作为独立变量
- limit(F)　　%对 x 求右趋于 a=0 的极限
- limit(F,x,a,'right')　　%对 x 求右趋于 a 的极限
- limit(F,x,a,'left')　　%对 x 求左趋于 a 的极限

例 3.32　求表达式极限。

```
>> syms x a t h;
>> limit(sin(x)/x)      %对 x 求右趋于 0 的极限
ans =
 1
 >> limit(1/x,x,0,'right') %对 x 求右趋于 0 的极限
ans =
Inf
>> limit(1/x,x,0,'left')   %对 x 求左趋于 0 的极限
ans =
-Inf
>> limit((sin(x+h)-sin(x))/h,h,0)   %对 x 求趋于 a=0 的极限
ans =
cos(x)
>> v = [(1 + a/x)^x, exp(-x)];     %对 x 求趋于无穷大的极限
limit(v,x,inf,'left')
ans =
[ exp(a),          0]
```

2. 符号级数

求符号表达式级数时，MATLAB 提供函数 symsum 和 taylor 来实现这一操作。

1）symsum 函数

符号表达式的级数求和常常采用函数 symsum，格式如下：

r = symsum(s,v,a,b)　%求符号表达式 s 在变量 v 取遍[a,b]中所有整数的和，v 省略则自动对自变量求和，a，b 同时省略时默认的求和的自变量区间为[0,v-1]

例 3.33　求级数 $1+\frac{1}{2^2}+\frac{1}{3^2}+...+\frac{1}{k^2}$ 的和。

```
>> syms x k
>> r=symsum(1/k^2,1,20)        %求级数前 20 项的和
r =
17299975731542641/10838475198270720
>> r=symsum(1/k^2,10,20)     %求级数前第 10 项至第 20 项的和
r =
3056206830982561/5419237599135360
```

2）taylor 函数

格式如下：

taylor(f,n,v)　%求表达式 f 进行泰勒级数展开 n 项，v 为自变量，默认泰勒展开式为 5 项。

例 3.34　求表达式 sin（x）和的 e^x 泰勒展开式。

```
>> syms x
>> r=taylor(sin(x),10)       %sinx 泰勒展开前 10 项
r =
x-1/6*x^3+1/120*x^5-1/5040*x^7+1/362880*x^9
>> r=taylor(exp(x),10)   %ex 泰勒展开前 10 项
r =
1+x+1/2*x^2+1/6*x^3+1/24*x^4+1/120*x^5+1/720*x^6+1/5040*x^7+1/40320*x^8
+1/362880*x^9
```

3．符号微分

（1）符号表达式的微分以下面 4 种形式利用函数 diff。

- diff(f)：传回 f 对预设独立变数的一次微分值。
- diff(f,'t')：传回 f 对独立变数 t 的一次微分值。
- diff(f, n)：传回 f 对预设独立变数的 n 次微分值。
- diff(f,'t',n)：传回 f 对独立变数 t 的 n 次微分值。

例 3.35 符号表达式微分。

```
>> f=sym('a*x^3+x^2-b*x-c')
f=
a*x^3+x^2-b*x-c
>> diff(f,'x')
ans=
3*a*x^2+2*x-b
>> diff(f,'a')
ans=
x^3
>> diff(f,2)
ans=
6*a*x+2
>> diff(f,'a',2)
ans=
0
```

函数 diff 也可对数组进行运算。如果 F 是符号向量或数组，diff(F)对数组内的各个元素进行微分。

例 3.36 符号数组微分。

```
>> F=sym('[a*x,b*x^2;c*x^3,d*s]')
F=
[     a*x, b*x^2]
[ c*x^3,     d*s]
>> diff(F)
ans=
[          a,     2*b*x]
[ 3*c*x^2,            0]
```



函数 diff 也用于在 MATLAB 中计算数值向量或矩阵的数值差分。对于一个数值向量或矩阵 M，diff(M)计算 M(2:m,:) M(1:m-1,:)的数值差分。

如果 diff 的表达式或可变参量是数值，MATLAB 就非常巧妙地计算其数值差分。如果参量是符号字符串或变量，MATLAB 就对其表达式进行微分。

（2）运用 Jacobian 函数对多元符号函数求导。

jacobian(f,v)函数用于计算数量或者向量 f 对于向量 v 的 jacobi（雅可比）矩阵，计算得到的结果的第 i 行第 j 列的数为 $df(i)/dv(j)$。当 f 为数量时，该函数返回的是 f 的梯度；v 为数量时，jacobian(f,v)等价于前面提到的 diff（f，v）。

例 3.37　利用 jacobian 函数对多元符号函数求导。

```
>> syms x y
>> a=[x^2+x*y+y^2;5*x*sin(x)*tan(y)]
a =
          x^2+x*y+y^2
 5*x*sin(x)*tan(y)
>> jacobian(a,[x,y])
ans =
[                              2*x+y,                   x+2*y]
[ 5*sin(x)*tan(y)+5*x*cos(x)*tan(y), 5*x*sin(x)*(1+tan(y)^2)]
```

4．符号积分

积分函数 int（f），其中 f 是一符号表达式，它力图求出另一符号表达式 F，使 diff(F)=f。正如研究微分学所了解的，积分有不定积分、定积分、旁积分和重积分等。一般来说，积分比微分要复杂得多。积分或逆求导不一定是以封闭形式存在的，或许存在但软件也许找不到，或者软件可明显地求解，但超过内存或时间限制。当 MATLAB 不能找到逆导数时，它将返回未经计算的命令。

相关的符号积分函数语法有以下 5 个。

- int(f)：传回 f 对预设独立变数的积分值。
- int(f,'t')：传回 f 对独立变数 t 的积分值。
- int(f,a,b)：传回 f 对预设独立变数 t 的积分值，积分区间为 $[a,b]$，a 和 b 为数值式。
- int(f,'t',a,b)：传回 f 对独立变数 t 的积分值，积分区间为 $[a,b]$，a 和 b 为数值式。
- int(f,'m','n')：传回 f 对预设变数 t 的积分值，积分区间为 $[m,n]$，m 和 n 为符号式。

下面分别就以上 5 种函数语法给出实例。

例 3.38　符号表达式积分。

```
>> int(sym('log(x)/exp(x^2)'))
Warning: Explicit integral could not be found.
ans=
int(log(x)/exp(x^2),x)
>> f=sym('sin(s+2*x)')
f=
sin(s+2*x)
>> int(f)
ans=
-1/2*cos(s+2*x)
>> int(f,'s')
ans=
```

```
-cos(s+2*x)
>> int(f,pi/2,pi)
ans=
-cos(x)
>> int(f,'s',pi/2,pi)
ans=
-sin(2*x)+cos(2*x)
>> int(f,'m','n')
ans=
1/2*cos(s+2*m)-1/2*cos(s+2*n)
```

正如函数 diff 一样，积分函数 int 对符号数组的每一个元素进行运算。

例 3.39 符号数组积分。

```
>> F=sym('[a*x,b*x^2;c*x^3,d*s]')
F=
[    a*x, b*x^2]
[ c*x^3,     d*s
>> diff(F)
ans=
[          a,     2*b*x]
[ 3*c*x^2,             0]
```

5. 符号积分变换

Fourier 变换、变换和 Z 变换在许多研究领域都有着十分重要的作用，特别是在信号处理和系统动态特性的研究中。Fourier（傅里叶）变换常常应用于连续系统，而 FFT（快速傅里叶变换）应用于离散系统；Laplace（拉普拉斯）变换常用于连续系统（微分方程），其离散模式的 Z 变换则常常应用于离散系统（差分方程）。

为此，MATLAB 提供了相应的函数来进行这些变换，下面将重点讨论这些变换的具体使用方法。

1）Fourier 变换及其反变换

时域中的 $f(x)$ 与它在频域中的 Fourier 变换存在如下关系：

$$f = f(x) \Rightarrow F = F(w) = \int_{-\infty}^{+\infty} f(x)\mathrm{e}^{-\mathrm{i}wx}\,\mathrm{d}x$$

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(w)\mathrm{e}^{\mathrm{i}wx}\,\mathrm{d}w$$

在 MATLAB 中分别由命令函数来完成此类变换，它们分别是 fourier 和 ifourier。对于 Fourier 变换其命令格式有如下 3 种。

- F = fourier(f): 对符号单值函数 f 中的默认变量 x（由命令 findsym 确定）计算 Fourier 变换形式。默认的输出结果 F 是变量 w 的函数：

$$f = f(x) \Rightarrow F = F(w) = \int_{-\infty}^{+\infty} f(x)\mathrm{e}^{-\mathrm{i}wx}\mathrm{d}x$$

  若 $f=f(w)$，则 fourier(f)返回变量为 t 的函数：$F=F(t)$。

- F = fourier(f,v)：对符号单值函数 f 中的指定变量 v 计算 Fourier 变换形式：

$$f = f(\oplus) \Rightarrow F = F(v) = \int_{-\infty}^{+\infty} f(x)\mathrm{e}^{-\mathrm{i}vx}\mathrm{d}x$$

- F = fourier(f,u,v)：令符号函数f为变量u的函数，而F为变量v的函数：

$$f = f(\oplus) \Rightarrow F = F(v) = \int_{-\infty}^{+\infty} f(u)\mathrm{e}^{-\mathrm{i}vu}\mathrm{d}x$$

例 3.40 符号函数的傅里叶变换。

```
>> syms x w u v
>> f=cos(x)*exp(x^2)
f =
cos(x)*exp(x^2)
>> F=fourier(f)
F =
1/2*fourier(exp(x^2)*exp(-i*x),x,w)+1/2*fourier(exp(x^2)*exp(i*x),x,w)
>> f1= x*exp(-abs(x))
f1 =
x*exp(-abs(x))
>> F1=fourier(f1,u)
F1 =
-4*i/(1+u^2)^2*u
>> syms x real
>> f2 = exp(-x^2*abs(v))*sin(v)/v
f2 =
exp(-x^2*abs(v))*sin(v)/v
>> F2=fourier(f2,v,u)
F2 =
atan((u+1)/x^2)-atan((u-1)/x^2)
```

傅里叶反变换使用 ifourier 命令来完成，其命令格式如下：

- f = ifourier(F,u)：输出参量$f=f(x)$为默认变量w的标量符号对象F的逆 Fourier 积分变换。即$F=F(w)$ → $f=f(x)$。若$F=F(x)$，ifourier(F)返回变量t的函数，即$F=F(x)$ → $f=f(t)$。逆 Fourier 积分变换定义为：

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(w)\mathrm{e}^{\mathrm{i}wx}\mathrm{d}w$$

- f = ifourier(F,u)：使函数f为变量u（u为标量符号对象）的函数：

$$f(u) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(w)\mathrm{e}^{\mathrm{i}wu}\mathrm{d}w$$

- f = ifourier(F,v,u)：使F为变量v的函数，f为变量u的函数：

$$f(u) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(v)\mathrm{e}^{\mathrm{i}wu}\mathrm{d}v$$

例 3.41 符号函数的傅里叶反变换。

```
>> syms x u v
>> f = cos(x)*exp(x^2);
>> F = ifourier(f)
F =
1/2*ifourier(exp(x^2)*exp(i*x),x,t)+1/2*ifourier(exp(x^2)*exp(-i*x),x,t)
>> f=2*exp(-abs(x))-1
f =
2*exp(-abs(x))-1
>> F=ifourier(f,u)
F =
```

```
-dirac(u)+2/(1+u^2)/pi
>> syms w real
>> f=exp(-w^2*abs(v))*sin(v)/v
f =
exp(-w^2*abs(v))*sin(v)/v
>> ifourier(f,v,u)
ans =
1/2*(-atan((u-1)/w^2)+atan((u+1)/w^2))/pi
```

2）Laplace 变换及其反变换

Laplace 变换定义为：$L(s)=\int_0^{+\infty}F(t)\mathrm{e}^{-st}\mathrm{d}t$，在 MATALB 中使用 laplace 命令来直接进行变换，其命令格式如下：

- laplace(F)：输出参量 $L=L(s)$为有默认符号自变量 t 的标量符号对象 F 的 Laplace 变换。即 $F=F(t)\ \rightarrow\ L=L(s)$。若 $F=F(s)$，则 fourier(F)返回变量为 t 的函数 L。
- laplace(F,t)：使函数 L 为变量 t（t 为标量符号自变量）的函数：

$$L(s)=\int_0^{+\infty}F(x)\mathrm{e}^{-tx}\,\mathrm{d}x$$

- laplace (F,w,z)：使 L 为变量 z 的函数，F 为变量 w 的函数：

$$L(z)=\int_0^{+\infty}F(w)\mathrm{e}^{-zw}\mathrm{d}w$$

具体使用实例如下所示。

例 3.42 符号函数的 Laplace 变换。

```
>> syms t
>> f=t^4
f =
t^4
>> laplace(f)
ans =
24/s^5
>> syms   a   t   x
>> f=exp(-a*t)
f =
exp(-a*t)
>> laplace(f,x)
ans =
1/(x+a)
```

逆 Laplace 变换定义为：$F(t)=\int_{c-\mathrm{i}\infty}^{c+\mathrm{i}\infty}L(s)\mathrm{e}^{st}\mathrm{d}t$，其 ilaplace 命令格式如下：

- ilaplace(L)：输出参量 $F=F(t)$为默认变量 s 的标量符号对象 L 的逆 Laplace 变换。即 $F=F(w)\ \rightarrow\ f=f(x)$。若 $L=L(t)$，则 ifourier(L)返回变量为 x 的函数 F。即 $F=F(x)\ \rightarrow\ f=f(t)$。逆 Laplace 变换定义为：$F(t)=\int_{c-\mathrm{i}\infty}^{c+\mathrm{i}\infty}L(s)\mathrm{e}^{st}\mathrm{d}t$，其中 c 为使函数 $L(s)$的所有的奇点位于直线 $s=c$ 左边的实数。
- ilaplace(L,y)：使函数 F 为变量 y（y 为标量符号对象）的函数：

$$F(y)=\int_{c-\mathrm{i}\infty}^{c+\mathrm{i}\infty}L(y)\mathrm{e}^{sy}\mathrm{d}s$$

- ilaplace(L,y,x)：使函数 F 为变量 x 的函数，L 为变量 y 的函数：

$$F(x)=\int_{c-i\infty}^{c+i\infty} L(y)e^{xy}dy$$

3）Z 变换及其反变换

函数 f 的 Z 变换定义为：$F(z)=\sum_{n=0}^{\infty}\frac{f(n)}{z^n}$，在 MATLAB 中使用命令 ztrans，其格式与上两节中使用的变换函数相似，主要有以下 3 种。

- ztrans(f)：对默认自变量为 n（就像由命令 findsym 确定的一样）的单值函数 f 计算 Z 变换。输出参量 F 为变量 z 的函数：$f=f(n)\rightarrow F=F(z)$。若函数 $f=f(z)$，则 ztrans(f)返回一个变量为 w 的函数：$f=f(z)\rightarrow F=F(w)$。
- ztrans(f,w)：用符号变量 w 代替默认的 z 作为函数 F 的自变量：

$$F(w)=\sum_{n=0}^{+\infty}\frac{f(n)}{w^n}$$

- ztrans(f,k,w)：对函数 f 指定的符号变量 k 计算 Z 变换：

$$F(w)=\sum_{n=0}^{+\infty}\frac{f(k)}{w^n}$$

逆 Z 变换定义为：$f(n)=\frac{1}{2\pi i}\oint_{|z|=R}F(z)z^{n-1}dz$，$n$ =1，2，3，…，最常见的 3 种 Z 反变换具体计算方法有：幂级数展开法、部分分式展开法和围线积分法。MATLAB 中采用围线积分法设计了求取 Z 反变换的 iztrans 指令。

例 3.43 符号函数的 Z 变换及其反变换。

下面将利用 Z 变换的相关函数求取序列 $f(n)=\sin(a*n)$ 的 Z 变换，指令如下：

```
>> syms a n w
>> f=sin(a*n)
f=
sin(a*n)
>> z_f=ztrans(f,w)
z_f=
-sin(a)*w/(2*w*cos(a)-w^2-1)
```

对于使用 ztrans 转换的序列，可以使用 iztrans 进行反变换来验算，例如：

```
>> iztrans(z_f)
ans =
sin(a*n)
```

可以看出，经过 iztrans 又变回了原来的序列 $f(n)$。

3.1.6 符号函数可视化

为了把符号运算得到的数值结果用图形显示出来，MATLAB 符号工具箱提供了许多符号作图函数来实现符号函数的可视化。

符号绘图函数可分为二维绘图和三维绘图，主要的绘图函数如表 3-6 所示。

表 3-6　常见符号绘图函数

函　数　名	功 能 说 明
ezplot	绘制符号表达式的自变量与对应的函数值的曲线
Ezplot3	绘制符号表达式的自变量与对应的函数值的三维曲线
Ezcontour	绘制等高线
Ezcontourf	绘制带填充颜色的等高线
ezmesh	绘制三维网线图
Ezmeshc	绘制带等高线的三维网线图
ezplar	绘制极坐标图
Ezsurf	绘制三维曲面图
ezsurfc	绘制带三维等高线的三维曲面图

下面举例说明符号绘图函数的应用。

1）ezcontourf 函数

格式如下：

- ezcontourf(fun)　%绘制二元符号函数 $f=f(x,y)$的等高线图，在不同的等高线之间填充不同颜色，函数默认的平面区域为[-2π<x<2π，-2π<y<2π]
- ezcontourf(fun,domain)　%函数在指定的定义域 domain 中绘制等高线图
- ezcontourf(...,n)　%函数在指定的 N×N 的栅格点内绘制等高线图，N 默认值为 60
- ezcontourf(axes_handle,...) %在指定的坐标轴 axes_handle 里面绘制等高线图
- h = ezcontourf(...)　%返回等高线的图像对象的句柄值

ezcontourf 函数自动添加标题和坐标轴标签。

例 3.44　运用函数 ezcontourf 绘制下面函数的等高线图。

$$f(x,y)=3(1-x)^2\mathrm{e}-x^2-(y+1)^2-10(\frac{x}{5}-x^3-y^5)\mathrm{e}-x^2-y^2-\frac{1}{3}\mathrm{e}-(x+1)^2-y^2$$

```
f = ['3*(1-x)^2*exp(-(x^2)-(y+1)^2)',...
        '- 10*(x/5 - x^3 - y^5)*exp(-x^2-y^2)',...
        '- 1/3*exp(-(x+1)^2 - y^2)'];
>> ezcontourf(f,[-3,3],49)
```

结果如 3-1 所示。

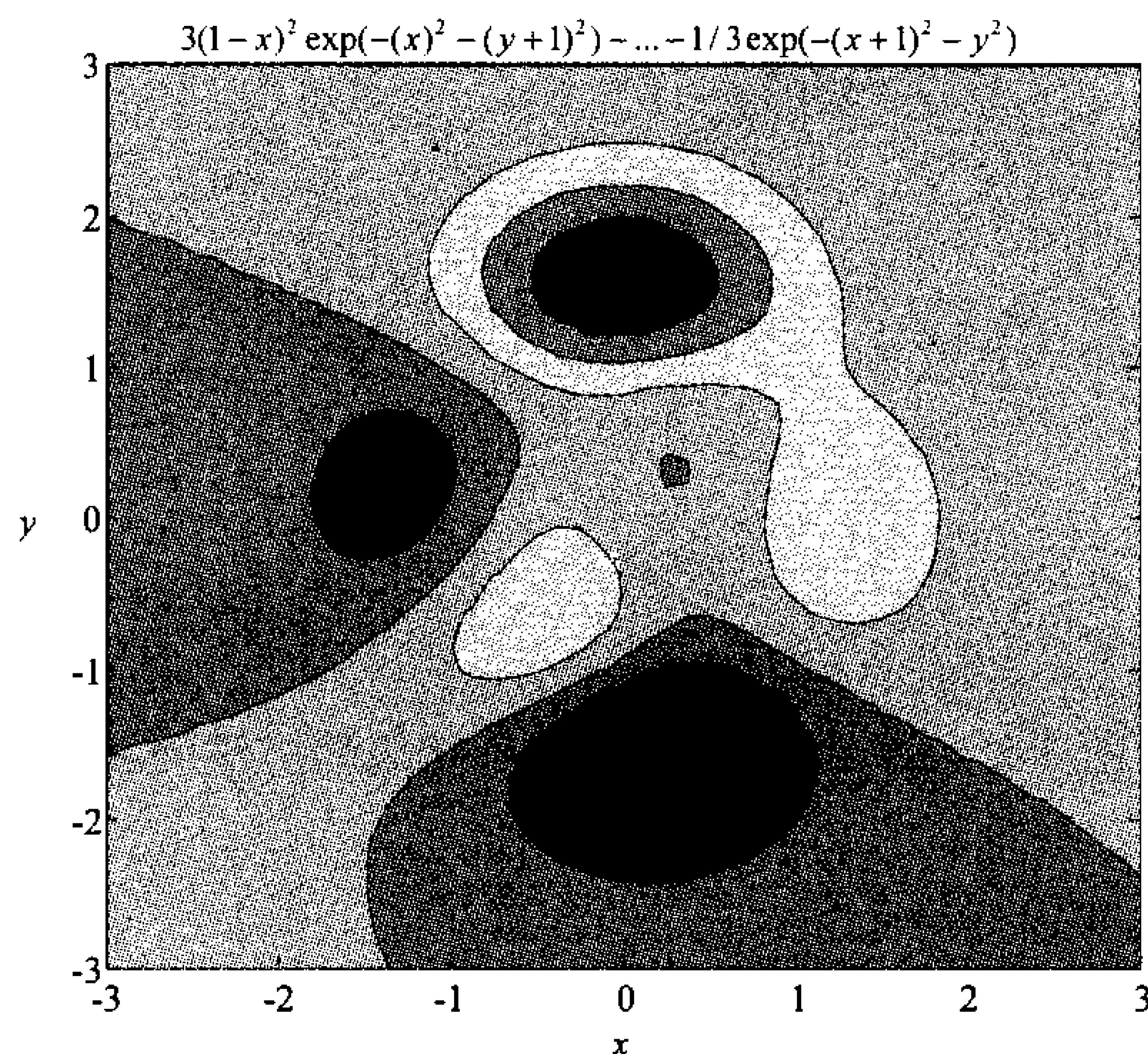


图 3-1　ezcontourf 函数绘制的不同颜色的等高线图

2）ezmeshc 函数

格式如下：

- ezmeshc(fun)　%绘制二元符号函数 z=f(x,y)的网格图，且同时在 xy 平面内显示其等高线图，函数默认的平面区域为[-2π<x<2π，-2π<y<2π]
- ezmeshc(fun,domain)　%函数在指定的定义域 domain 中绘制网格图，domain 可以是四维向量[xmin,xmax,ymin,ymax]或者二维向量[a,b]
- ezmeshc(funx,funy,funz)　%函数在指定的矩形定义域范围[-2π < s <2π, -2π < t < 2π]中绘制参数形式函数 x=x(s,t)、y=y(s,t)和 z=z(s,t)的二元函数 z=f(x,y)网格图
- ezmeshc(funx,funy,funz,[smin,smax,tmin,tmax])　%函数在指定的矩形定义域范围[smin < s <smax, tmin < t < tmax]中绘制参数形式函数 x=x(s,t)、y=y(s,t)和 z=z(s,t)的二元函数 z=f(x,y)网格图
- ezmeshc(funx,funy,funz,[min,max])　%函数在指定的定义域 [min,tmax]绘制二元函数 z=f(x,y)网格图
- ezmeshc(...,n)　%函数在指定的 $N\times N$ 的栅格点内绘制网格图，N 默认值为 60
- ezmeshc(...,'circ')　%函数在一圆形区域内绘制网格图
- ezmesh(axes_handle,...)　%函数在指定的坐标轴 axes_handle 里面绘制网格图
- h = ezmeshc(...)　%函数返回网格图的图像对象的句柄值

例 3.45　运用函数 ezmeshc 绘制下面函数的网格图，且 x，y 的取值范围为-5<x<5，-2*pi<y<2*pi。

$$f(x,y)=\frac{x}{1+x^2+y^2}$$

```
ezmeshc('y/(1 + x^2 + y^2)',[-5,5,-2*pi,2*pi])
```

结果如 3-2 所示。

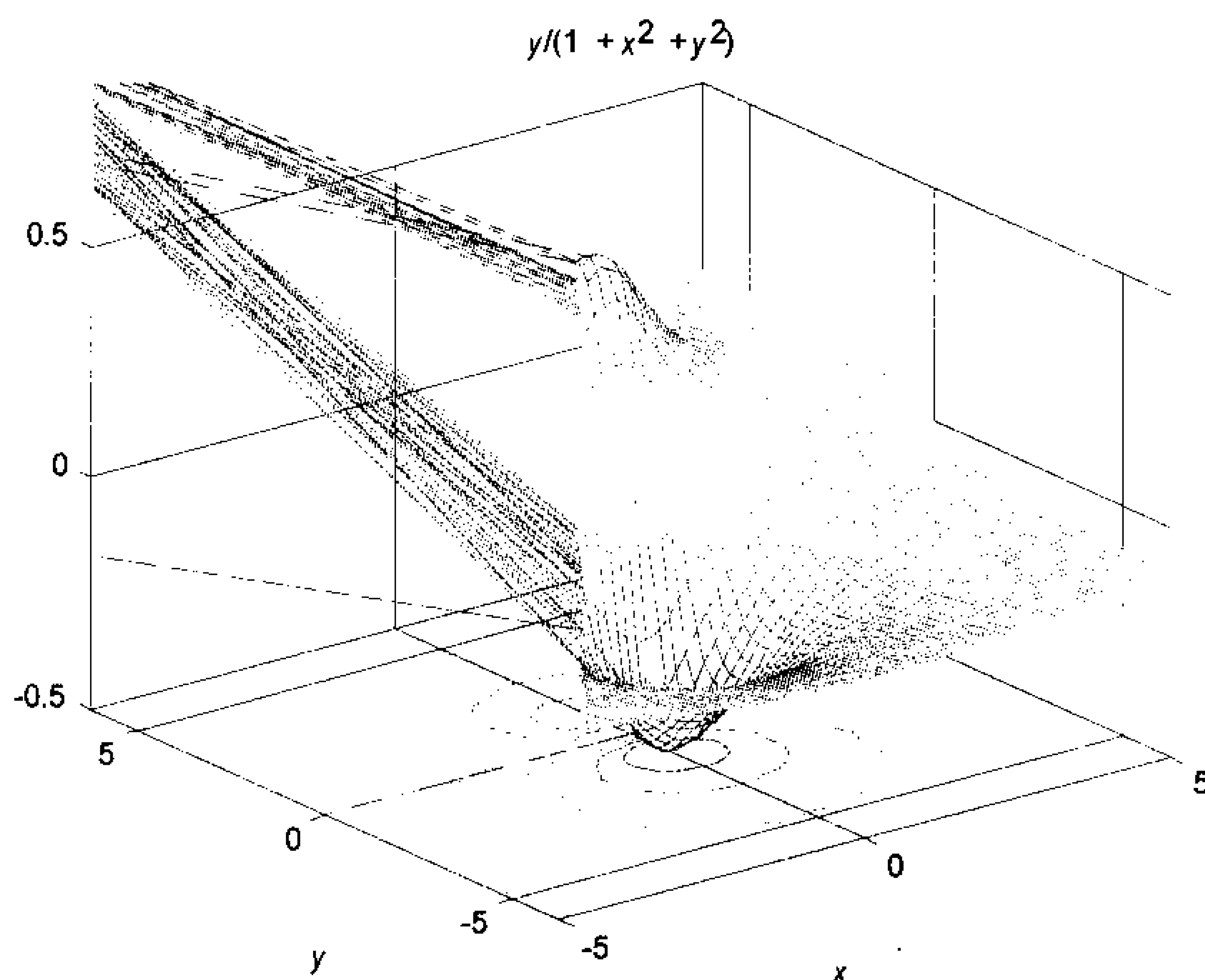


图 3-2　ezmeshc 绘制的带等高线的三维网线图

3）ezplot 函数

格式如下：

- ezplot(fun)　%绘制显式符号函数 y=f(x)在范围[-π<x<π]上的函数 f(x)的图形，隐式

符号函数 y=f(x，y)在范围[-2π<x<2π，-2π<y<2π]上的函数 f(x,y)的图形

- ezplot(fun,[min,max]) %函数在指定的定义域[min,max]中绘制图形
- ezplot(fun2) % 绘制函数 fun2(x,y)= 0 在默认定义域[-2π < x <2π, -2π < y < 2π]上的图形
- ezplot(fun2,[xmin,xmax,ymin,ymax]) %绘制函数 fun2(x,y)= 0 在指定定义域[xmin<x<xmax,ymin<y<ymax]上的图形
- ezplot(fun2,[min,max]) %绘制函数 fun2(x,y)=0 在指定定义域[min<x<max, min<y<max]上的图形
- ezplot(funx,funy) %绘制由平面曲线 funx(t),funy(t)定义的参数在默认定义域[0<t<2π]上的图形
- ezplot(funx,funy,[tmin,tmax]) %在定义域[tmin,tmax]内绘制参数函数 funx(t),funy(t)的图形
- ezplot(...,figure_handle) %在由 figure_handle 定义的图形窗口内绘制出给定函数在特定定义域内的图形
- ezplot(axes_handle,...) %在指定的坐标轴 axes_handle 绘制图形
- h = ezplot(...) %返回曲线对象的图形句柄值

例 3.46 运用函数 ezplot 绘制函数 $x^2 - y^4 = 0$，且 x，$y \in [-2\pi, 2\pi]$的图形。

```
ezplot('x^2-y^4')
```

结果如 3-3 所示。

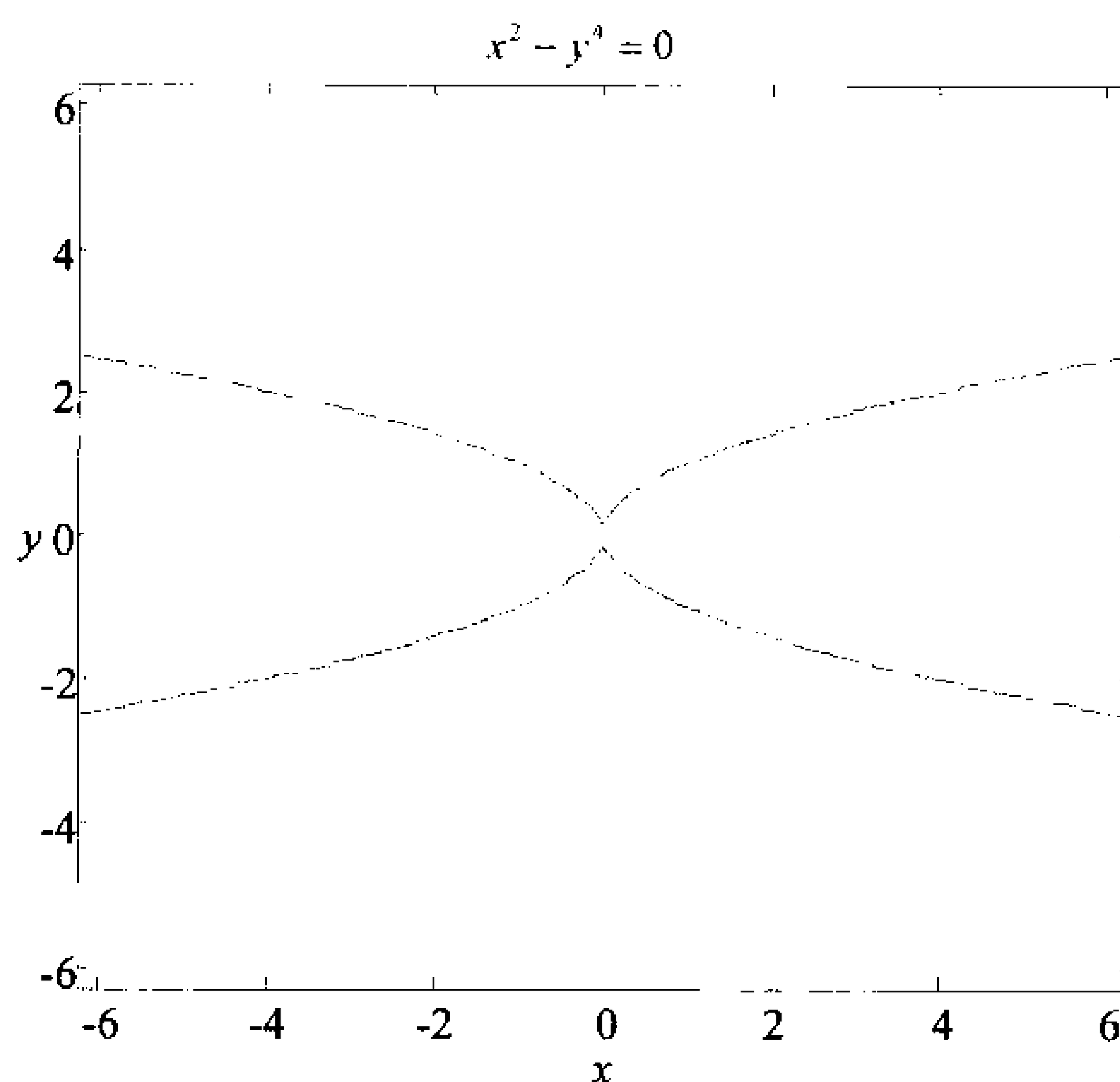


图 3-3 函数 ezplot 绘制的极坐标图形

表格中其他函数的使用方法类似，请读者参考符号函数手册。

3.1.7 符号方程求解

本节将主要介绍用 MATLAB 所具有的符号工具来求解符号方程。有些工具已经在前面介绍过，本节将进行更加详细的讲解。

1）求解代数方程

在前面已经看到，MATLAB 具有求解符号表达式的工具。如果表达式不是一个方程式（不含等号），则在求解之前指令 solve 将表达式置为 0。这里所讲的代数方程包括线性、非线性和超越方程等，求解指令都是 solve。其常用指令格式如下：

- g = solve(eq)：输入参量 eq 是符号表达式或字符串。若 eq 是一符号表达式 x^2 -2*x-1 或一个没有等号的字符串"x^2-2*x-1"，则 solve(eq)对方程 eq 中的默认变量（由命令 findsym(eq)确定的变量）求解方程 eq=0。若输出参量 g 为单一变量，则对于有多重解的非线性方程，g 为一行向量。
- g = solve(eq,var)：对符号表达式或没有等号的字符串 eq 指定的变量 var 求解方程 eq(var)=0。
- g = solve(eq1,eq2,⋯,eqn)：输入参量 eq1,eq2,⋯,eqn 可以是符号表达式或字符串。该命令对方程组 eq1,eq2,⋯,eqn 中由命令 findsym 确定的 n 个变量如 x1,x2,⋯,xn 求解。若 g 为一单个变量，则 g 为一包含 n 个解的结构；若 g 为有 n 个变量的向量，则分别返回结果给相应的变量。
- g = solve(eq1,eq2,⋯,eqn,var1,var2,⋯,varn)：对方程组 eq1,eq2,⋯,eqn 中指定的 n 个变量，如 var1,var2,⋯,varn 求解。

例 3.47 代数方程的求解。

```
>> solve('a*x^2+b*x+c')
ans=
-1/2*(b-(b^2-4*a*c)^(1/2))/a
 -1/2*(b+(b^2-4*a*c)^(1/2))/a
```

结果是符号向量，其元素是方程的两个解。如果想对非默认 x 变量求解，solve 必须指定变量。

```
>> solve('a*x^2+b*x+c','b')   %solve for b
ans=
-(a*x^2+c)/x
```

带有等号的符号方程也可以求解。

```
>> f=solve('cos(x)=sin(x)')          %solve for x
f=
1/4*pi
>> t=solve('tan(2*x)=sin(x)')
t=
    atan(1/2*(-2*3^(1/2))^(1/2),1/2+1/2*3^(1/2))
   atan(-1/2*(-2*3^(1/2))^(1/2),1/2+1/2*3^(1/2))
  atan(1/2*2^(1/2)*3^(1/4)/(1/2-1/2*3^(1/2)))+pi
  -atan(1/2*2^(1/2)*3^(1/4)/(1/2-1/2*3^(1/2)))-pi
                                               0
                                              pi
```

在求解周期函数方程时，有无穷多的解。在这种情况下，solve 对解的搜索范围限制在接近于零的有限范围内，并返回非唯一解的子集。

如果不能求得符号解，就计算可变精度解。

```
>> x=solve('exp(x)=tan(x)')
x=
   1.3063269404230792361743566584407
```

例 3.48 小敏想去看电影，她从小猪存钱罐倒出硬币并清点，然后她发现：

（1）1角的硬币数加上1分和5分的硬币总数的一半等于5角的硬币数。

（2）1分的硬币数比5分、1角以及5角的硬币总数多10。

（3）5角和1角的硬币总数等于1分的硬币数加上1/4的5分的硬币数。

（4）5角的硬币数和1分的硬币数比5分的硬币数加上8倍的1角的硬币数多1。

如果电影票价为4.00元，爆米花为1.50元，糖棒为5角，她有足够的钱去买这三样东西吗？

首先，根据以上给出的信息列出一组线性方程，假如 p，n，d 和 q 分别表示1分、5分、1角和5角的硬币数。

$$\begin{cases} d+\dfrac{n+p}{2}=q \qquad p=n+d+q-10 \qquad q+d=p+\dfrac{n}{4} \\ q+p=n+8d-1 \end{cases}$$

然后，建立MATLAB符号方程并对变量求解。

```
>> equ1='d+(n+p)/2=q';
>> equ2='p=n+d+q-10';
>> equ3='q+d=p+n/4';
>> equ4='q+p=n+8*d-1';
>> [pennies,nickles,dimes,aquarters]=solve(equ1,equ2,equ3,equ4,'p,n,d,q')
pennies =
            3
nickles =
            8
dimes =
           16
aquarters =
             15
```

所以，得出小敏有3枚1分的硬币，8枚5分的硬币，16枚1角的硬币，15枚5角的硬币。

```
>> money=.01*pennies+.05*nickles+.10*dimes+.5*aquarters
money=
          953/100
```

这就意味着她有足够的钱去买电影票、爆米花和糖棒并剩余2.36元。

2）求解微分方程

从数值计算的角度看，与数值问题求解相比，微分方程边值问题的求解显得更为复杂和困难。MATLAB提供了功能强大的工具，可以帮助求解微分方程。函数dsovle计算常微分方程的符号解，其指令格式如下：

```
r = dsolve('eq1,eq2,…','cond1,cond2,…','v')
```

对给定的常微分方程（组）eq1,eq2，…中指定的符号自变量 v，与给定的边界条件和初始条件cond1,cond2,…求符号解（即解析解）r；若没有指定变量 v，则默认变量为 t；在微分方程（组）的表达式eq中，大写字母D表示对自变量（设为 x）的微分算子：D=d/dx，D2=d2/dx2，…。微分算子D后面的字母则表示为因变量，即待求解的未知函数。初始和边界条件由字符串表示：$y(a)=b$，D$y(c)=d$，D2$y(e)=f$ 等，分别表示 $y(x)|_{x=a}=b$，$y'(x)|_{x=c}=d$，$y''(x)|_{x=e}=f$。若边界条件少于方程（组）的阶数，则返回的结果 r 中会出

现任意常数 C1，C2，…。dsolve 命令最多可以接受 12 个输入参量（包括方程组与定解条件个数，当然，我们可以做到输入的方程个数多于 12 个，只要将多个方程置于一个字符串内即可）。若没有给定输出参量，则在命令窗口显示解列表。若该命令找不到解析解，则返回一个警告信息，同时返回一个空的 sym 对象。这时，用户可以用命令 ode23 或 ode45 求解方程组的数值解。因为我们要求解微分方程，需要用一种方法将微分包含在表达式中。所以，dsovle 句法与大多数其他函数有些不同，用字母 D 来表示求微分，D2，D3 等表示多重复微分，并以此来设定方程。任何 D 后所跟的字母为因变量。方程 $d^2y/dx^2=0$ 用符号表达式 D2y=0 来表示。独立变量可以指定或由 symvar 规则选定为默认。

例 3.49　一阶方程 $\frac{dy}{dx}=1+y^2$ 的通解如下：

```
>> dsolve('Dy=1+y^2')
ans=
tan(t+C1)
```

其中，C1 是积分常数。求解初值 $y(0)=1$ 的同一个方程就可以产生：

```
>> dsolve('Dy=1+y^2 ','y(0)=1')          % add an initial condition
y=
tan(t+1/4*pi)
```

独立变量可用如下形式指定：

```
>> dsolve('Dy=1+y^2','y(0)=1','v')       % find solution to dy/dv
ans=
tan(v+1/4*pi)
```

让我们举一个二阶微分方程的例子，如例 3.50 所示。

例 3.50　设方程及其两个初始条件如下：

$$\frac{d^2y}{dx^2}=\cos(2x)-y\quad\frac{dy}{dx}(0)=0\quad y(0)=1$$

求解该方程的解。

```
>> y=dsolve(' D2y=cos(2*x)-y ', ' Dy(0)=0 ', ' y(0)=1 ')
y=
cos(t)*(1-cos(2*x))+cos(2*x)
>> y=simple(y)
y=
cos(t)*(1-cos(2*x))+cos(2*x)
```

通常，要求解的微分方程含有一阶以上的项，并以下述形式表示：

$$\frac{d^2y}{dx^2}-2\frac{dy}{dx}-3y=0$$

通解为：

```
>> y=dsolve( 'D2y-2Dy-3*y=0 ')
y=
exp(3^(1/2)*t)*C2+exp(-3^(1/2)*t)*C1-2/3
```

加上初始条件：$y(0)=0$ 和 $y(1)=1$，可得到：

```
>> y=dsolve('D2y-2Dy-3*y=0','y(0)=0','y(1)=1')
y=
1/3*exp(3^(1/2)*t)*(5*exp(3^(1/2))-2)/(exp(3^(1/2))^2-1)+1/3*exp(-3^(1/2)*t)*exp(3^(1/2))*(2*exp
(3^(1/2))-5)/(exp(3^(1/2))^2-1)-2/3
>> y=simple(y)              ....%this looks like a candidate for simplification
```

```
y=
1/3*(5*exp(3^(1/2)*(t+1))-2*exp(3^(1/2)*t)+2*exp(-3^(1/2)*(t-2))-5*exp(-3^(1/2)*(t-1))-2*exp(2*3
^(1/2))+2)/(exp(2*3^(1/2))-1)
>> pretty(y)                          %pretty it up
                 1/2                  1/2                1/2
   1/3 (5 exp(3      (t + 1)) - 2 exp(3     t) + 2 exp(-3      (t - 2))
                      1/2                 1/2         /        1/2
          - 5 exp(-3      (t - 1)) - 2 exp(2 3    ) + 2)  /   (exp(2 3    ) - 1)
                                                      /
```

现在来绘制感兴趣的区域内的结果：

```
>> ezplot(y,[-6 2])
```

结果如图 3-4 所示。

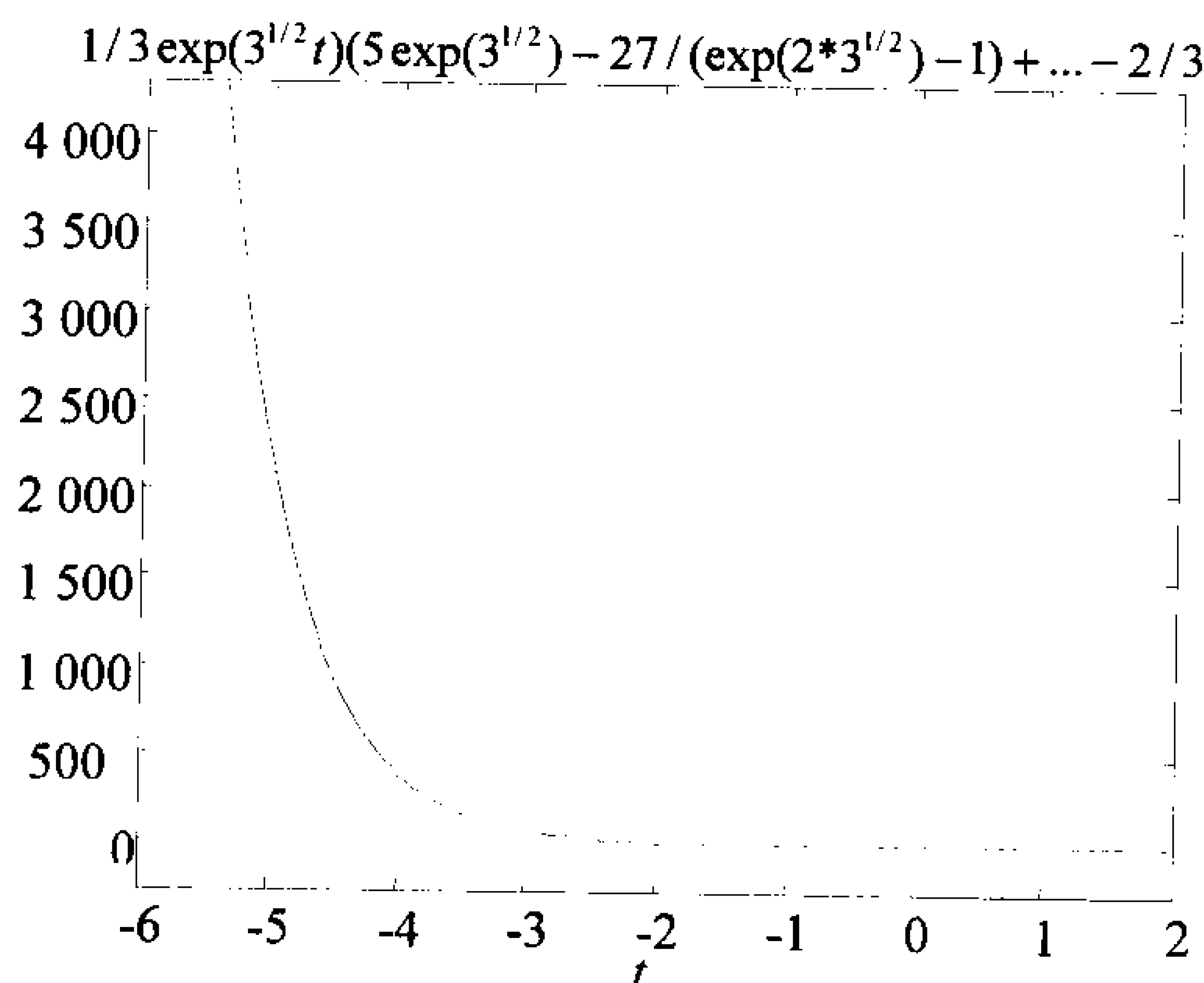


图 3-4 函数曲线图

函数 dsolve 也可同时处理若干个微分方程式，即微分方程组，如例 3.51 所示。

例 3.51 设有如下两个线性一阶方程。

$$\frac{du}{dx}=6u+3v \qquad \frac{dv}{dx}=-3u+9v$$

求其通解。

```
>> [u,v]=dsolve('Du=6*u+3*v','Dv=3*u+9*v')
u =
-1/2*C1*exp(3/2*(5+5^(1/2))*t)+1/2*C1*exp(3/2*(5+5^(1/2))*t)*5^(1/2)-1/2*C2*exp(-3/2*(-5+5^(
1/2))*t)-1/2*C2*exp(-3/2*(-5+5^(1/2))*t)*5^(1/2)
v =
C1*exp(3/2*(5+5^(1/2))*t)+C2*exp(-3/2*(-5+5^(1/2))*t)
```

加上初始条件：$u(0)=0$ 和 $v(0)=1$，可以得到：

```
>> [u,v]=dsolve('Du=6*u+3*v','Dv=3*u+9*v','u(0)=0','v(0)=1')
u =
-1/2*(1/10*5^(1/2)+1/2)*exp(3/2*(5+5^(1/2))*t)+1/2*(1/10*5^(1/2)+1/2)*exp(3/2*(5+5^(1/2))*t)*5^(1
/2)-1/2*(-1/10*5^(1/2)+1/2)*exp(-3/2*(-5+5^(1/2))*t)-1/2*(-1/10*5^(1/2)+1/2)*exp(-3/2*(-5+5^(1/2))*t)*5^(1/2)
v =
(1/10*5^(1/2)+1/2)*exp(3/2*(5+5^(1/2))*t)+(-1/10*5^(1/2)+1/2)*exp(-3/2*(-5+5^(1/2))*t)
```

3.1.8 Maple 函数

前面介绍了符号计算最基础的内容，只涉及用简单的几条指令就能解决的问题。但是，事实上，用户通常除了会遇到这些简单的指令就能解决的问题以外，还会遇到许多更为复杂的问题，此时读者可以使用 Maple 提供的各种函数来解决。Maple 具有很强大的符号计算功能和丰富的经典应用数学函数，这些资源以库的形式提供给 MATLAB，由于不是 M 文件，因而不能直接在 MATLAB 中使用，MATLAB 为此提供了专用的函数作为接口，通过这些函数访问 Maple 的内核，从而可以很容易地调用 Maple 的绝大多数功能。

MATLAB 提供了如下 5 个实现 MATLAB 和 Maple 之间交互的指令。

- mfun：对 Maple 中若干经典特殊函数实施数值计算。
- mfunlist：能被 mfun 计算的库函数及其调用方法。
- mhelp：查阅 Malpe 中的库函数及其调用方法。
- maple：进入 Maple 工作空间计算，结果送回 MATLAB 工作空间。
- procread：把按 Maple 格式编写的源程序读入 Maple 工作空间。

利用符号数学工具箱中的 Maple 函数可以直接访问 Maple 中的任何函数。这个函数采用符号对象、字符串、双精度数作为输入。返回与输入相对应的符号对象、字符串和双精度数。也可以使用 Maple 函数调试用户编写的符号数学程序。

Maple 函数调用格式如下。

- r = maple('statement')：将参数命令 statement 传递给 Maple 内核，且返回计算结果。在必要时，可以在参量 statement 后面加上分号（;）。
- r = maple('function',arg1,arg2,…)：该命令接受任何带引号的函数名'function'与相关的输入参量 arg1，arg2，…在必要时，要将输入参量转换成符号表达式。若输入参量为 syms，则 maple 返回一个 sym，否则返回一个类型为 char 的结果。
- [r, status] = maple(…)：有条件地返回警告/错误信息。当语句能顺利执行时，则 r 为计算结果，status 为 0；若语句不能通过执行，则 r 为相应的警告/错误信息，而 status 为一个正整数。
- maple('traceon')，maple traceon，maple trace on：将显示所有的后面的 Maple 语句与其相应的结果。
- maple('traceoff')，maple traceoff，maple trace off：将关闭上面的操作特性。

例 3.52 下面将利用 Maple 函数来求解一个递推方程的通解，先给定如下方程：

$$f(n) = -2f(n-1) - f(n-2)$$

下面利用两种常用的调用格式访问 Maple 函数：

```
>> maple('rsolve(f(n)=-2*f(n-1)-f(n-2),f(k));')
ans =
(f(1)+2*f(0))*(-1)^k+(-f(0)-f(1))*(k+1)*(-1)^k
>> maple('rsolve','f(n)=-2*f(n-1)-f(n-2)','f(k)')
ans =
(f(1)+2*f(0))*(-1)^k+(-f(0)-f(1))*(k+1)*(-1)^k
```

读者需要注意的是，例子中用到的 rsolve 函数是 Maple 提供的，MATLAB 没有提供直接的这样的函数用于求解递推方程。rsolve 的调用格式，可以通过使用命令 mhelp 来获得。

在符号数学工具箱中，有 50 多个经典的应用数学的特殊函数可供使用，其中绝大部分在 MATLAB 中不能直接求解。这些函数可以通过 mfun 函数访问，其用法为：

mfun(function,Parl,Par2,Par3,Par4)

计算指定的 Maple 软件中已知的数学函数 function 的数值。每一参量 Par 为该函数相应的具体数值。用户可以输入 4 个参量。最后指定的参量可以是矩阵，通常对应于 x。其他参量的位数取决于该函数规定的范围。

符号数学工具箱中提供了函数 mfunlist，可以列出 Maple 中的特殊数学函数名称及其功能，这些函数的用法等更加详细的信息可以用命令“mhelp+函数名”来获取。

注意前面介绍的函数 maple 也可以调用这些数学函数，它和函数 mfun 的区别在于计算方式不同。函数 maple 采用符号计算，计算结果是符号对象或字符型数据；而函数 mfun 采用 16 位精度的数值计算，结果是双精度型数值，出现奇异值则返回为 NaN。

采用 mfun 可以调用下面这些特殊数学函数，如表 3-7 所示。

表 3-7　特殊函数表

函数名	定义	Mfun 名	参量说明
Bernoulli 数与多项式	生成函数：$\frac{e^{xt}}{e^t-1}=\sum_{n=0}^{\infty}B_n(x)*\frac{t^{n-1}}{n!}$	Bernoulli(n) Bernoulli(n,t)	$n\geqslant0$ $0<\|t\|<2\pi$
Bessel 函数	BesselI, BesselJ：第 1 类 Bessel 函数 BesselK, BesselY：第 2 类 Bessel 函数	BesselJ(v,x) BesselY(v,x) BesselI(v,x) BesselK(v,x)	v 为实数
Beta 函数	$B(x,y)=\frac{\Gamma(x)\times\Gamma(y)}{\Gamma(x+y)}$	Beta(x,y)	
二项式系数	$\binom{m}{n}=\frac{m!}{n!(m-n)!}=\frac{\Gamma(m+1)}{\Gamma(n+1)\times\Gamma(m+n+1)}$	Binomial(m,n)	
完全椭圆积分	第一、二、三类 Legendre 完全椭圆积分	LegendreKc(k) LegendreEc(k) LegendrePic(a,k)	a 为任意实数 $-\infty<a<\infty$ k 为任意实数 $0<k<1$
带余模的完全椭圆积分	与余模相关的第一、二、三类 Legendre 完全椭圆积分	LegendreKc1(k) LegendreEc1(k) LegendrePic1(a,k)	a 为任意实数 $-\infty<a<\infty$ k 为任意实数 $0<k<1$
余差函数与它的累积分	$\mathrm{Erfc}(z)=\sqrt{2}\int_z^{+\infty}e^{-t^2}\,dt=1-\mathrm{erf}(z)$ $\mathrm{erfc}(n,z)=\int_z^{+\infty}\mathrm{erfc}(n-1,z)\,dt$	erfc(z) erfc(n,z)	$n>0$
Dawson 积分	$F(x)=e^{-x^2}\int_0^x e^{-t^2}\,dt$	dawson(x)	
Ψ-函数	$\psi(x)=\frac{d}{dx}\ln\Gamma(x)$	Psi(x)	
二重对数积分	$f(x)=\int_1^x\frac{\ln t}{1-t}\,dt$	dilog(x)	$x>1$
误差函数	$\mathrm{erf}(z)=\sqrt{2}\int_0^z e^{-t^2}\,dt$	erf(z)	
Euler 数与多项式	生成 Euler 数的函数：$\frac{1}{\mathrm{ch}\,t}=\sum_{n=0}^{\infty}E_n\times\frac{t^n}{n!}$	euler(n) euler(n,z)	$n\geqslant0$ $\|t\|<\pi/2$
指数积分	$E_i(n,z)=\int_1^{+\infty}\frac{e^{-zt}}{t^n}\,dt$ $E_i(x)=PV-\int_{-\infty}^{x}\frac{e^t}{t}\,dt$	Ei(n,z) Ei(x)	$n\geqslant0$ real(z)>0

（续表）

函 数 名	定 义	Mfun 名	参 量 说 明
Fresnel 正弦与余弦积分	$C(x)=\int_0^x \cos(\frac{\pi}{2}t^2)\mathrm{d}t$ $S(x)=\int_0^x \sin(\frac{\pi}{2}t^2)\mathrm{d}t$	FresnelC(x) FresnelS(x)	
Γ-函数	$\Gamma(z)=\int_0^{+\infty} t^{z-1}\mathrm{e}^{-t}\mathrm{d}t$	GAMMA(z)	
调和函数	$h(n)=\sum_{k=1}^{n}\frac{1}{k}=\Psi(n+1)+\gamma$	harmonic(n)	$n>0$
双曲正弦与余弦积分	$\mathrm{Shi}(z)=\int_0^z \frac{\sin ht}{t}\mathrm{d}t$ $\mathrm{Chi}(z)=\gamma+\ln(z)+\int_0^z \frac{\cos ht-1}{t}\mathrm{d}t$	Shi(z) Chi(z)	
广义超几何函数	$F(n,d,z)=\sum_{k=0}^{\infty}\frac{\prod_{i=1}^{j}\frac{\Gamma(n_i+k)}{\Gamma(n_i)}z^k}{\prod_{i=1}^{m}\frac{\Gamma(d_i+k)}{\Gamma_i(di)}k!}$	hypergeom(n,d,x) 其中 $n=[n1,n2,\cdots]$ $d=[d1,d2,\cdots]$	$n1$，$n2$，… 为实数 $d1$，$d2$，… 为非负实数
不完全椭圆积分	第一、二、三类不完全 Legendre 完全椭圆积分	LegendreF(x,k) LegendreE(x,k) LegendrePi(x,a,k)	$0<x\leqslant+\infty$，a 为实数 $-\infty<a<+\infty$，k 为实数 $0<k<1$
不完全Γ-函数	$\Gamma(a,z)=\int_z^{+\infty}\mathrm{e}^{-t}t^{a-1}\mathrm{d}t$	GAMMA(z1,z2)	
Γ-函数的对数	$\ln\Gamma(z)=\ln(\Gamma(z))$	lnGAMMA(z)	
对数积分	$\mathrm{Li}(x)=\mathrm{PV}\left(\int_0^x \frac{\mathrm{d}t}{\ln t}\right)=\mathrm{Ei}(\ln(x))$	Li(x)	$x>1$
Γ多项式函数	其中$\Psi(z)$为Γ-函数 $\psi^{(n)}(z)=\frac{\mathrm{d}^n\psi(z)}{\mathrm{d}z^n}$	Psi(n,z)	$n\geqslant0$
移位正弦积分	$\mathrm{Ssi}(z)=\mathrm{Si}(z)-\pi/2$	Ssi(z)	

对于上面的特殊函数 function，用户可以通过命令 mhelp function 获得更多的帮助信息。

例 3.53 特殊函数值计算的操作过程。首先，利用命令 mfunlist 获得需要的函数名。

```
>> mfunlist
MFUNLIST Special functions for MFUN.
    The following special functions are listed in alphabetical order
    according to the third column. n denotes an integer argument,
    x denotes a real argument, and z denotes a complex argument. For
    more detailed descriptions of the functions, including any
    argument restrictions, see the Reference Manual, or use MHELP.
  Bernoulli       n       Bernoulli Numbers
  Bernoulli       n,z     Bernoulli Polynomials
  BesselI         x1,x    Bessel Function of the First Kind
  BesselJ         x1,x    Bessel Function of the First Kind
  BesselK         x1,x    Bessel Function of the Second Kind
  BesselY         x1,x    Bessel Function of the Second Kind
  Beta            z1,z2   Beta Function
  binomial        x1,x2   Binomial Coefficients
  EllipticF -     z,k     Incomplete Elliptic Integral, First Kind
  EllipticK -     k       Complete Elliptic Integral, First Kind
  EllipticCK -    k       Complementary Complete Integral, First Kind
  EllipticE -     k       Complete Elliptic Integrals, Second Kind
```

```
EllipticE -     z,k         Incomplete Elliptic Integrals, Second Kind
EllipticCE -    k           Complementary Complete Elliptic Integral, Second Kind
EllipticPi -    nu,k        Complete Elliptic Integrals, Third Kind
EllipticPi -    z,nu,k      Incomplete Elliptic Integrals, Third Kind
EllipticCPi -   nu,k        Complementary Complete Elliptic Integral, Third Kind
erfc            z           Complementary Error Function
erfc            n,z         Complementary Error Function's Iterated Integrals
Ci              z           Cosine Integral
dawson          x           Dawson's Integral
Psi             z           Digamma Function
dilog           x           Dilogarithm Integral
erf             z           Error Function
euler           n           Euler Numbers
euler           n,z         Euler Polynomials
Ei              x           Exponential Integral
Ei              n,z         Exponential Integral
FresnelC        x           Fresnel Cosine Integral
FresnelS        x           Fresnel Sine Integral
GAMMA           z           Gamma Function
harmonic        n           Harmonic Function
Chi             z           Hyperbolic Cosine Integral
Shi             z           Hyperbolic Sine Integral
GAMMA           z1,z2       Incomplete Gamma Function
W               z           Lambert's W Function
W               n,z         Lambert's W Function
lnGAMMA         z           Logarithm of the Gamma function
Li              x           Logarithmic Integral
Psi             n,z         Polygamma Function
Ssi             z           Shifted Sine Integral
Si              z           Sine Integral
Zeta            z           (Riemann) Zeta Function
Zeta            n,z         (Riemann) Zeta Function
Zeta            n,z,x       (Riemann) Zeta Function
            Orthogonal Polynomials (Extended Symbolic Math Toolbox only)
T               n,x         Chebyshev of the First Kind
U               n,x         Chebyshev of the Second Kind
G               n,x1,x      Gegenbauer
H               n,x         Hermite
P               n,x1,x2,x   Jacobi
L               n,x         Laguerre
L               n,x1,x      Generalized Laguerre
P               n,x         Legendre
    See also mfun, mhelp.
    Reference page in Help browser
       doc mfunlist
```

假设需要使用 FresnelS 函数，那么接下来可以使用 mhelp 命令：

```
>> mhelp FresnelS
FresnelC - The Fresnel Cosine Integral
FresnelS - The Fresnel Sine Integral
Fresnelf, Fresnelg - The Fresnel Auxiliary Functions
```

```
Calling Sequence
        FresnelC(x)
        FresnelS(x)
        Fresnelg(x)
        Fresnelf(x)
Parameters
        x - algebraic expression
Description
- The Fresnel cosine integral is defined as follows:
        FresnelC(x) =   int(cos(Pi/2*t^2), t=0..x);
- The Fresnel sine integral is defined as follows:
        FresnelS(x) =   int(sin(Pi/2*t^2), t=0..x);
- The Fresnel auxiliary functions are defined as follows:
        Fresnelf(x) = (1/2 - FresnelS(x))*cos(Pi/2*x^2) -
                                    (1/2 - FresnelC(x))*sin(Pi/2*x^2)
        Fresnelg(x) = (1/2 - FresnelC(x))*cos(Pi/2*x^2) +
                                    (1/2 - FresnelS(x))*sin(Pi/2*x^2)
Examples
> FresnelS(infinity);
                                      1/2
> FresnelC(1);
                                   FresnelC(1)
> evalf(%);
                                   0.7798934004
> Fresnelf(1.0);
                                   0.2798934004
See Also
erf, dawson, inifcns
```

接下来就可以直接利用 mfun 命令来使用 FresnelS 函数了。

```
>> mfun('FresnelS',0:10)
ans =
  Columns 1 through 8
         0    0.4383    0.3434    0.4963    0.4205    0.4992    0.4470    0.4997
  Columns 9 through 11
    0.4602    0.4999    0.4682
```

3.2 关系运算及逻辑运算

除了拥有强大的矩阵数学运算功能和前面介绍的符号运算外，MATLAB R2007 同样拥有功能强大的关系运算和逻辑运算。在执行关系及逻辑运算时，MATLAB 将输入的不为 0 的数值都视为真（True），而为 0 的数值则视为否（False）。同时，判断为真者以 1 表示，而判断为否者以 0 表示，各个运算元须用在两个大小相同的数组或是矩阵中的比较。

这一节将简要阐述关系运算符与逻辑运算符、运算符优先级、关系和逻辑函数以及其相关应用。

3.2.1 关系运算符与逻辑运算符

1. 关系操作符

MATLAB R2007 中的关系运算符主要有 6 个操作符，如表 3-8 所示。

表 3-8 MATLAB 中的关系操作符

关系操作符	操作符说明	对应的函数
==	等于	Eq（A,B）
~=	不大于	Ne（A,B）
<	小于	Lt（A,B）
>	大于	Gt（A,B）
<=	小于等于	Le（A,B）
>=	大于等于	Ge（A,B）



关系运算函数中 ***A*** 或 ***B*** 一般为数组或者矩阵。在 MATLAB 中进行关系运算时，要求数组或矩阵大小一致，且返回的结果依然是大小和原矩阵一致的矩阵。

例 3.54 比较两矩阵的大小。

```
>> A=[1 5 9;3 4 7;2 6 8];
>> B=magic(3);
>> C=gt(A,B)
C =
     0     1     1
     0     0     0
     0     0     1
```

2. 逻辑运算符

MATLAB R2007 中的逻辑运算符与其他语言一样主要有 3 种：逻辑与、逻辑或和逻辑非，其特点和关系运算符相似。

相应的符号和运算函数如表 3-9 所示。

表 3-9 MATLAB 中的逻辑操作符

逻辑操作符	操作符说明	对应的函数
&	逻辑与	And（A,B）
|	逻辑或	Or（A,B）
~	逻辑非	Nor（A,B）

例 3.55

```
>> A=[1 5 0;3 0 7;2 6 8];
>> B=magic(3);
>> C=and(A,B)
C =
```

```
1    1    0
1    0    1
1    1    1
```

3.2.2 运算符优先级

在 MATLAB R2007 中各种运算符的优先级依次降低，如表 3-10 所示。

表 3-10 运算符的优先级

各种运算符的优先级依次降低
'（矩阵转置）、^（矩阵幂）、.'（数组转置）、.^（数组幂）
~（逻辑非）
（乘）、/（左除）、\.（右除）、.（点乘）./（左点除）、.\（右点除）
+（加）、－（减）
:（冒号）、<、>、<=、>=、~=
&（逻辑与）
|（逻辑或）
&&（先决与）
||（先决或）

运算符的优先级依次降低

3.2.3 关系和逻辑函数

在 MATLAB 中提供了许多关系和逻辑运算判断的函数（也有资料称为测试函数），现给出程序设计中经常使用的一些关系逻辑函数，如表 3-11 所示。

表 3-11 测试函数

函 数 名	函 数 说 明
All(A)	判断 *A* 的列元素是否全非 0，全非 0 返回 1，否则返回 0
Any(A)	判断 *A* 的列元素中是否有非 0 元素，有则返回 1，否则返回 0
Isqual(A,B)	判断 *A*、*B* 对应的元素是否全部相等，是返回 1，否则返回 0
Isempty（A）	判断 *A* 是否为空矩阵，是返回 1，否则返回 0
Isfinite(A)	判断 *A* 各元素是否有限，是返回 1，否则返回 0
Isinf(A)	判断 *A* 各元素是否无穷，是返回 1，否则返回 0
Isnan(A)	判断 *A* 各元素是否为 NaN，是返回 1，否则返回 0
Isnumeric(A)	判断 *A* 各元素是否全为数值型数，是返回 1
Isreal(A)	判断 *A* 各元素是否全为实数，是返回 1，否则返回 0
Isprime（A）	判断 *A* 各元素是否为质数，是返回 1，否则返回 0
Isspace（A）	判断参量 *A* 是否为空格字符，是返回 1，否则返回 0
Isstr（A）or ischar（A）	判断参量 *A* 是否为一个字符串，是返回 1，否则返回 0
Isstudent（A）	判断 MATLAB 是否为学生版，是返回 1，否则返回 0
Isunix（A）	判断系统是否为 UNIX，是返回 1，否则返回 0
Isvms（A）	判断系统是否为 vms，是返回 1，否则返回 0
Find（A）	查询 A 中非 0 元素的下标和值

3.2.4 关系和逻辑运算实例

例 3.56 利用数组的关系和逻辑运算求半波整流图形。

```
>> x=linspace(0,3*pi);           %构建新数组
   y=sin(x);                      %计算正弦曲线数据
   x1=(x<pi)|(x>2*pi);            %运用关系和逻辑运算得出 x1 新的数据区间
   y1=x1.*y;
   plot(x,y1)
```

根据数据绘出图形，如图 3-5 所示。

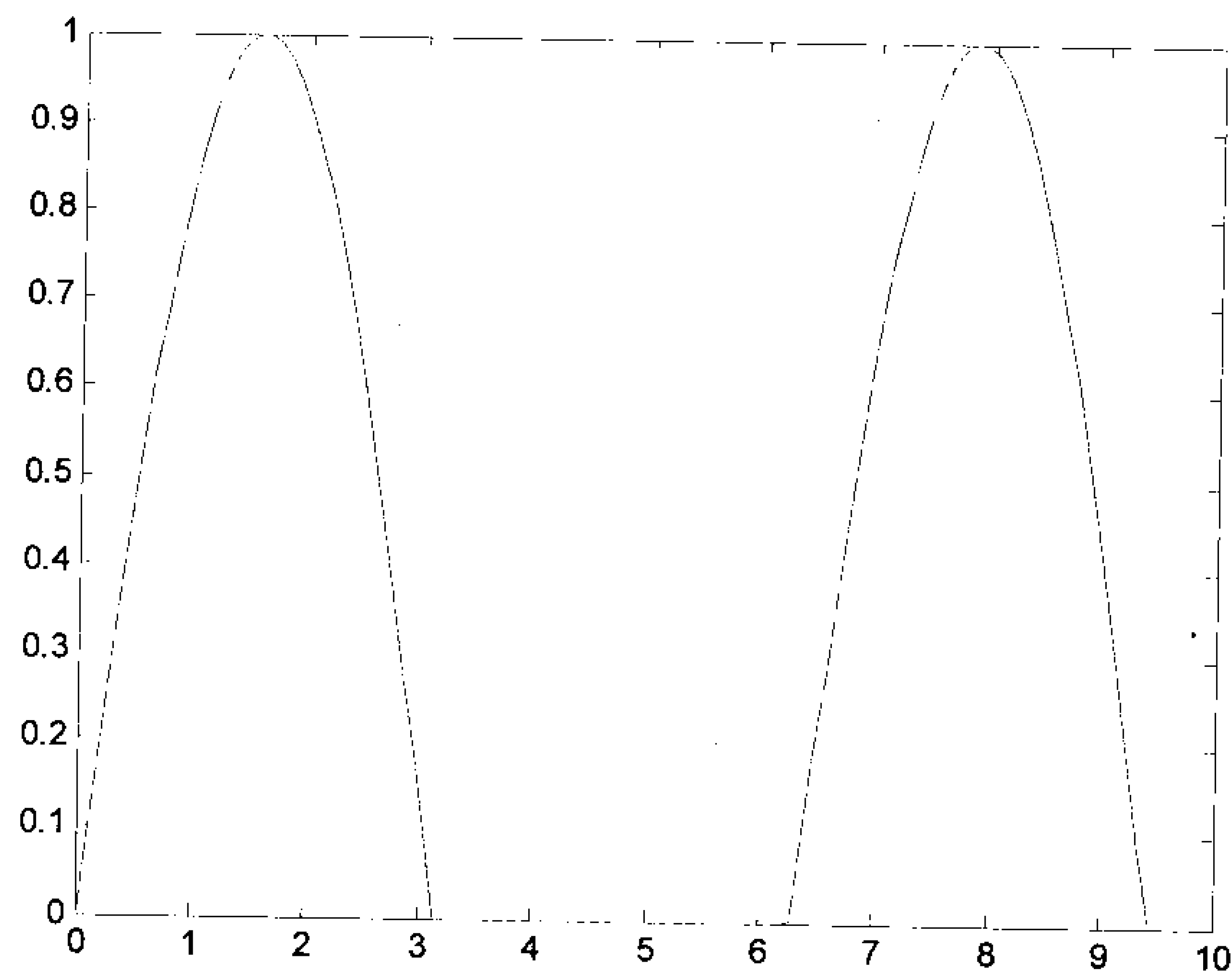


图 3-5 半波整流波形

3.3 多项式及其运算

多项式运算是数学中最基本的运算之一，在许多学科里面都有着非常广泛的应用。MATLAB R2007 提供了许多多项式运算函数，如多项式的求值、求根、多项式的微积分运算、曲线拟合、插值以及部分分式展开等，常用的一些函数如表 3-12 所示。

表 3-12 常用的多项式操作函数

函 数	功 能 描 述
Conv	多项式相乘、卷积
Deconv	多项式相除、反卷积
Poly	用多项式的根求多项式系数
Polyer	多项式求导
Polyfit	多项式拟合
Polyval	多项式求值
Polyvalm	矩阵多项式评价
Residue	部分分式展开（残差运算）
Roots	多项式求根

3.3.1 多项式求值

MATLAB 提供的求值函数为 polyval 和 polyvalm，有些资料也称为多项式评价。

格式如下：

Polyval（p,x）

Polyvalm（p,x）

其中，x 可以是复数，也可以是矩阵或者数组。两个函数的区别是，前者是按照数组运算规则来计算多项式的值，而后者 x 必须为方阵，且是按照矩阵运算规则来计算多项式的值。

例 3.57 求多项式 $f(x)=3x^3+5x^2+7x-21$ 在区间 [-2,10]均匀取 100 个点的 $f(x)$ 值，并画出曲线图形。

```
x=linspace(-2, 10);
>> p=[3 5 7 -21]
p =
    1     4     -7     -10
>> v=polyval(p,x)
>> plot(x , v),title(' 3x^3+5x^2+7x-21 '),xlabel(' x ')
```

所以，得到了多项式 $f(x)$ 值，如图 3-6 所示。

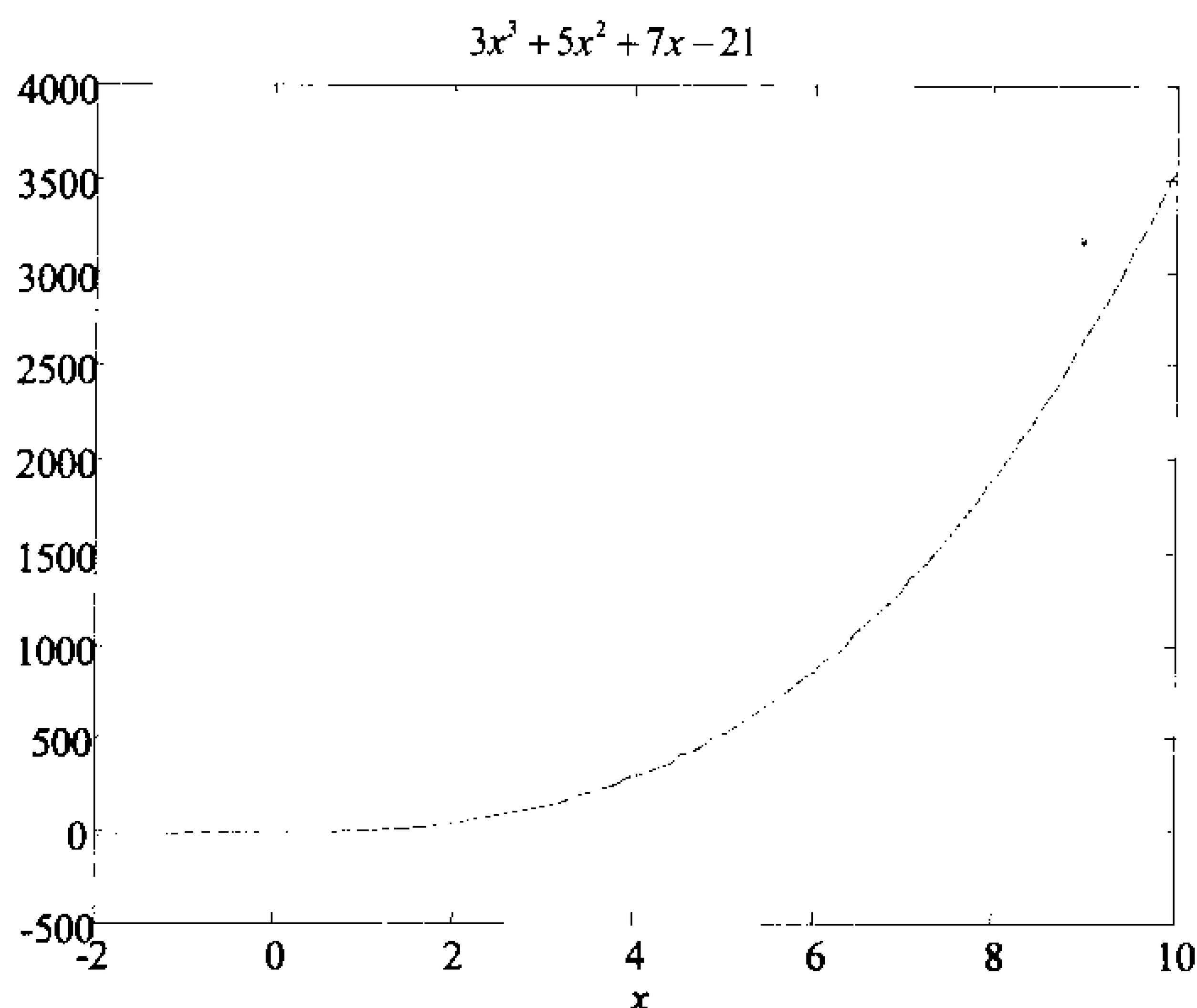


图 3-6 多项式求值

3.3.2 多项式求根

多项式求根运算在数学计算中非常常见。在 MATLAB R2007 中提供函数 roots 来求根，可以通过函数 poly 由多项式的根得出多项式系数。它们为一对互为逆函数，在校验排序、比例化、圆整导致的错误时经常用到它们。

格式如下：

r=roots(p);

p=poly(r)

例 3.58 求多项式 $f(x)=x^3+6x^2+11x+6$ 的根以及依据根得出多项式的系数。

```
>> p=[1 6 11 6];
>> r=roots(p)
r =
    -3.0000
```

```
    -2.0000
    -1.0000
>> p=poly([-3 -2 -1])
p =
     1     6    11     6
```

3.3.3 部分分式展开

在信号处理和控制系统的分析应用中常常需要将分母多项式和分子多项式构成的传递函数进行部分分式展开。从表 2-7 中可知运用函数 residue()来实现部分分式展开操作。

$$\frac{b(s)}{a(s)}=\frac{r_1}{s-p_1}+\frac{r_2}{s-p_2}+\cdots+\frac{r_n}{s-p_n}+k(s)$$

格式如下：

[r,p,k] = residue(b,a)　　%b，a 分别为分子和分母多项式系数的行向量，r 为留数行向量

[b,a] = residue(r,p,k)　　% p 为极点行向量，k 为直项行向量

例 3.59　求表达式 $f(s)=\dfrac{5s^3+3s^2-2s+7}{-4s^3+8s+3}$ 的部分分式展开式子。

```
>> b=[5 3 -2 7];
>> a=[-4 0 8 3];
>> [r,p,k]=residue(b,a)
r =
    -1.4167
    -0.6653
     1.3320
p =
     1.5737
    -1.1644
    -0.4093
k =
    -1.2500
>> [b,a] = residue(r,p,k)
b =
    -1.2500    -0.7500     0.5000    -1.7500
a =
     1.0000    -0.0000    -2.0000    -0.7500
```

所以，部分分式展开的表达式为 $f(s)=\dfrac{-1.25s^3-0.75s^2+0.50s-1.75}{s^3-2.00s-0.75}$。

3.3.4 多项式乘除

多项式的乘法运算和除法运算在 MATLAB 中分别通过函数 conv()和 deconv()来实现，同时卷积和反卷积运算也使用这两个函数。

格式如下：

w = conv(u,v)

[q,r] = deconv(v,u)

例 3.60　求多项式 $f(x)=5x^3+6x^2+3x+9$ 和 $g(x)=7x^4+8x^3+3x^2+10x+2$ 的乘积。

```
>> u=[5 6 3 9];
>> v=[7 8 3 10 2];
>> w= conv(u,v)
w =
     35     82     84    155    151     69     96     18
```

这样，得到两个多项式相乘后的结果为：

$$T(x)=35x^7+82x^6+84x^5+155x^4+151x^3+69x^2+96x+18$$

例 3.61 求例 3.60 中两个多项式的商。

```
>> [q,r]=deconv(u,v)       %用 u 除以 v
q =
     0
r =
     5     6     3     9
>> [q,r]=deconv(v,u)       %用 v 除以 u
q =
     1.4000    -0.0800
r =
    -0.0000     0.0000    -0.7200    -2.3600     2.7200
>> conv(q,u)+r             %验算得到的结果
ans =
      7.0000     8.0000     3.0000    10.0000     2.0000
```

从例子中可以看出，在多项式除法运算时，不一定能够整除，可能会有余子式 r。

3.3.5 多项式的微积分

在 MATLAB R2007 中有专门函数 polyder()来做多项式的微分运算，而未提供积分运算的函数，一般通过式子[p./length(p):-1:1,k]来进行积分运算。

格式如下：

m=polyder(p)

例 3.62 求多项式 $f(x)=5x^3+6x^2+3x+9$ 微分。

```
>> p=[5 6 3 9];
>>m=polyder(p)
m =
     15     12     3
```

所以，微分后得到的多项式为 $g(x)=15x^2+12x+3$。

例 3.63 对例 3.62 中得到的多项式 $g(x)=15x^2+12x+3$ 求积分。

```
>> p=[5 6 3 9];
>> m=polyder(p)
m =
     15     12     3
>> s=length(m):-1:1
s =
      3     2     1
>> p=[m./s,0]
p =
      5     6     3     0
```

所以，我们得到多项式 $g(x)=15x^2+12x+3$ 的积分是 $f(x)=5x^3+6x^2+3x+C$。

第4章　MATLAB 高级绘图技术

图形具有直观形象、清晰易懂的优点，能给人以视觉冲击。用图形来显示数学计算的结果，可以让用户更加容易理解和接受，更能够增加说服力。MATLAB 提供了极其丰富的绘图功能，它具有数百个绘图和图形操作方面的函数，不仅可以绘制二维、三维甚至更高维图形，还可以通过对图形的线型、平面、色彩、光线和视角等要素的控制，使绘制的图形尽善尽美。

MATLAB 绘图的一般步骤包括：（1）输入图形的数据信息；（2）调用绘图函数进行绘图；（3）设置图形属性，包括坐标轴标注、颜色设置、线型设置等，以达到较为理想的表现形式，这个步骤也可以和（2）合并，通过对绘图命令增加后缀形式直接实现；（4）输出或打印文件图形。其中（2）和（3）是绘图技术所要掌握的重点，而图形属性和图形对象的处理方法是绘图技术的难点，读者应该加强这方面的学习和理解。

本章首先结合大量的工程示例讲述了二维、三维及更高维绘图功能的实现方法，内容涉及一般工程应用中遇到的几乎所有类型的图形，如曲面图、面积图、直方图、等高线图，以及三维曲线图、立体切片图、向量图、瀑布图等，并介绍了坐标轴标注、图形标题、曲线标注（如有多条曲线）及重要数值标注等；接着重点讲述了 MATLAB 图形绘制的高级技术——图形对象操作方法，使读者掌握 MATLAB 图形的高级属性更改；最后本章介绍了 MATLAB 动画的制作方法。

本章主要内容：

- 二维图形绘制
- 三维图形绘制
- 图形色彩处理
- 句柄式图形
- 图像显示技术
- 动画制作

4.1　二维图形绘制

在 MATLAB 图形绘制过程中，我们最先涉及的就是二维曲线图形的绘制。其中不仅仅是绘制基本的二维图形，还有相当多的特殊二维图形的绘制，如条形图、直方图、扇形图等。同时，对绘制图形的文字标注能够帮助用户和其他阅读者更好地了解图形本身的含义。本节介绍基本 MATLAB 二维图形的绘制方法，并依据完整的步骤来说明一个图形产生的流程，以便将分析的数据立即以图形形式来识别。

4.1.1 基本二维绘图

MATLAB 中提供了一些非常实用的基本二维绘图函数，帮助用户绘制一连串的向量资料，如表 4-1 所示。

表 4-1 基本二维绘图函数

函数名称	用法	函数名称	用法
plot	二维曲线图	area	面积图
polar	二维极坐标图	pie	扇形图
loglog	双轴对数坐标图	scatter	散点图
semilogx	X 轴对数刻度二维绘图	hist	柱形图
semilogy	Y 轴对数刻度二维绘图	errorbar	误差图
bar	垂直条形图	stem	火柴杆图
barh	水平条形图	feather	羽毛图
quiver	向量图	commet	彗星图
rose	玫瑰花图	contour	等值线图
stairs	阶梯图	compass	罗盘图

在二维曲线绘图指令中，最重要、最基本的指令是 plot，其他许多特殊绘图指令都是以它为基础而形成的。作为绘制线性坐标平面图形的函数 plot，对于不同的输入参数，该函数用不同的形式可以实现不同的功能，我们在这里将分别进行介绍。该命令调用格式如表 4-2 所示。

表 4-2 plot 函数调用格式

调用格式	说明
plot(x,y)	绘制具有同长度 n 的向量组（x,y）的图形。其中 x 为[1,2,3,⋯,n]；y 可以是长度为 n 的实数向量，也可以是 n 行的实数矩阵，每列绘制一条曲线；如果 y 是一个复向量，则绘制实部曲线和虚部曲线
plot(x1,y1,...)	绘制多个同长度向量组（x_i,y_i）的图形。如果其中某对 x、y 是矩阵，则按 x、y 匹配的方向配对绘制曲线
plot(x1,y1,S,...)	同 plot(*x1*,*y1*,...)，但每一组向量由参数 S 确定曲线的线和颜色，可以同时使用 3 个或 2 个参数
plot(...,'ProName','ProVal',...)	对所有用 plot 函数创建的图形进行属性设置
h=plot(...)	返回函数 plot 绘制曲线的句柄属性值，每一条曲线给出一个属性值向量

例程 4.1 plot 命令示例

```
>> x=linspace(0,2*pi);    %linspace 用以指定在一特定范围内均匀取点
>> y=sin(x);
>> plot(x,y,x,(y-2),'rd');
```

最后一个命令表示画两组数据，'rd'是对后一组曲线的属性更改，为红色菱形，在本章 4.1.4 节有详细介绍。输出结果如图 4-1 所示。

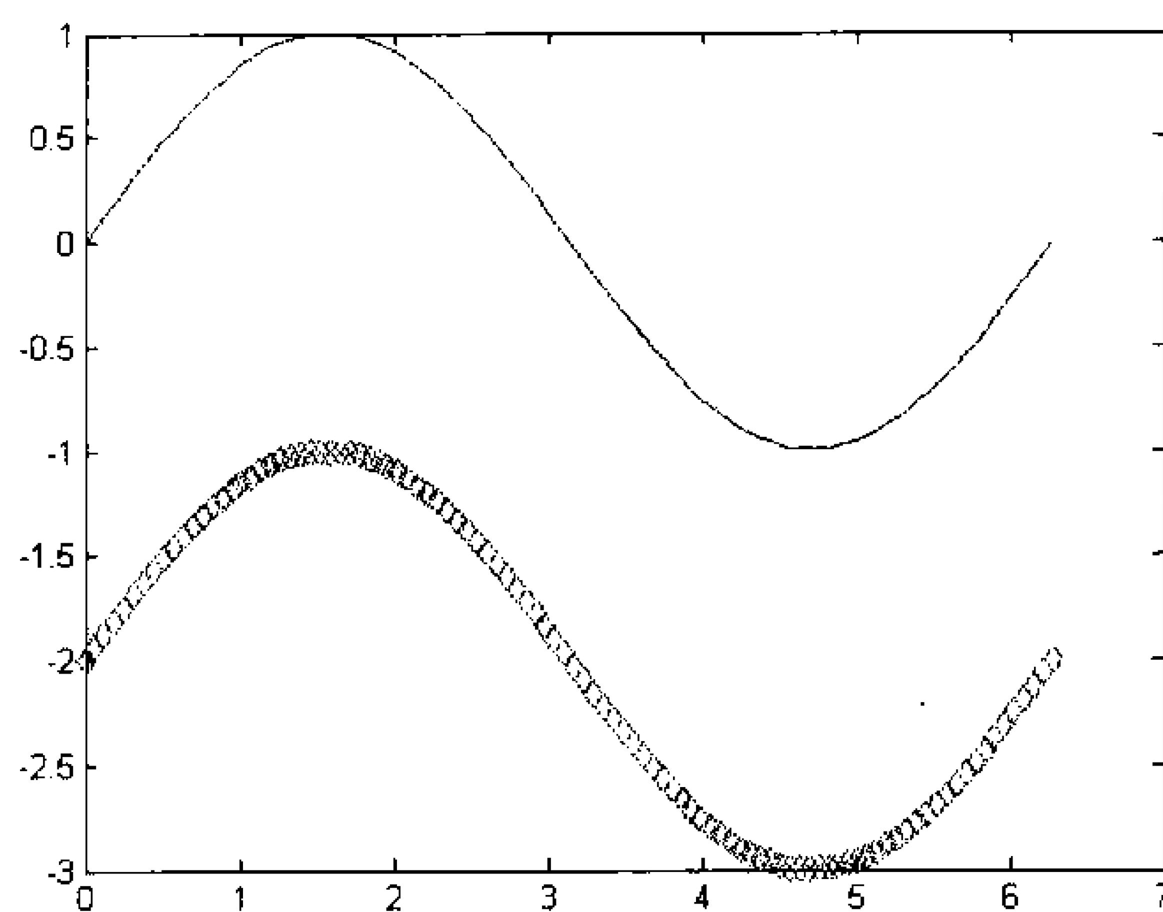


图 4-1　使用 plot 命令绘制多条数据结果

当指令 plot 的输入变量 s 由线性符、色彩符中各选一个符号组合而成时，plot 指令就使用所选定的线，从“微观上”把那些给定的离散数据逐个用“直线”连接起来，生成“宏观上”的曲线。

如例程 4.2 所示，画出曲线 $y = \mathrm{e}^{-\frac{t}{3}}\cos(10t)$ 及其包络线 $y_0 = \mathrm{e}^{-\frac{t}{3}}$。$T$ 的取值范围是[0，4π]。

例程 4.2　绘制曲线及其包络线

```
t=linspace(0,4*pi);
y0=exp(-t/3);
y=exp(-t/3).*cos(10*t);
p=plot(t,y,t,y0,':r',t,-y0,':r');
grid on      %打开格网
set(p(1),'LineWidth',2);      %设置曲线宽度
legend('e^{-t/3}cos(10t)','e^{-t/3}');      %图形标注
title('e^{-t/3}cos(10t)','Fontsize',14);      %设置标题并设置文字大小
ylabel('e^{-t/3}','Fontsize',14);      %标注 Y 轴标题
```

如图 4-2 所示为例程 4.2 的绘制结果。

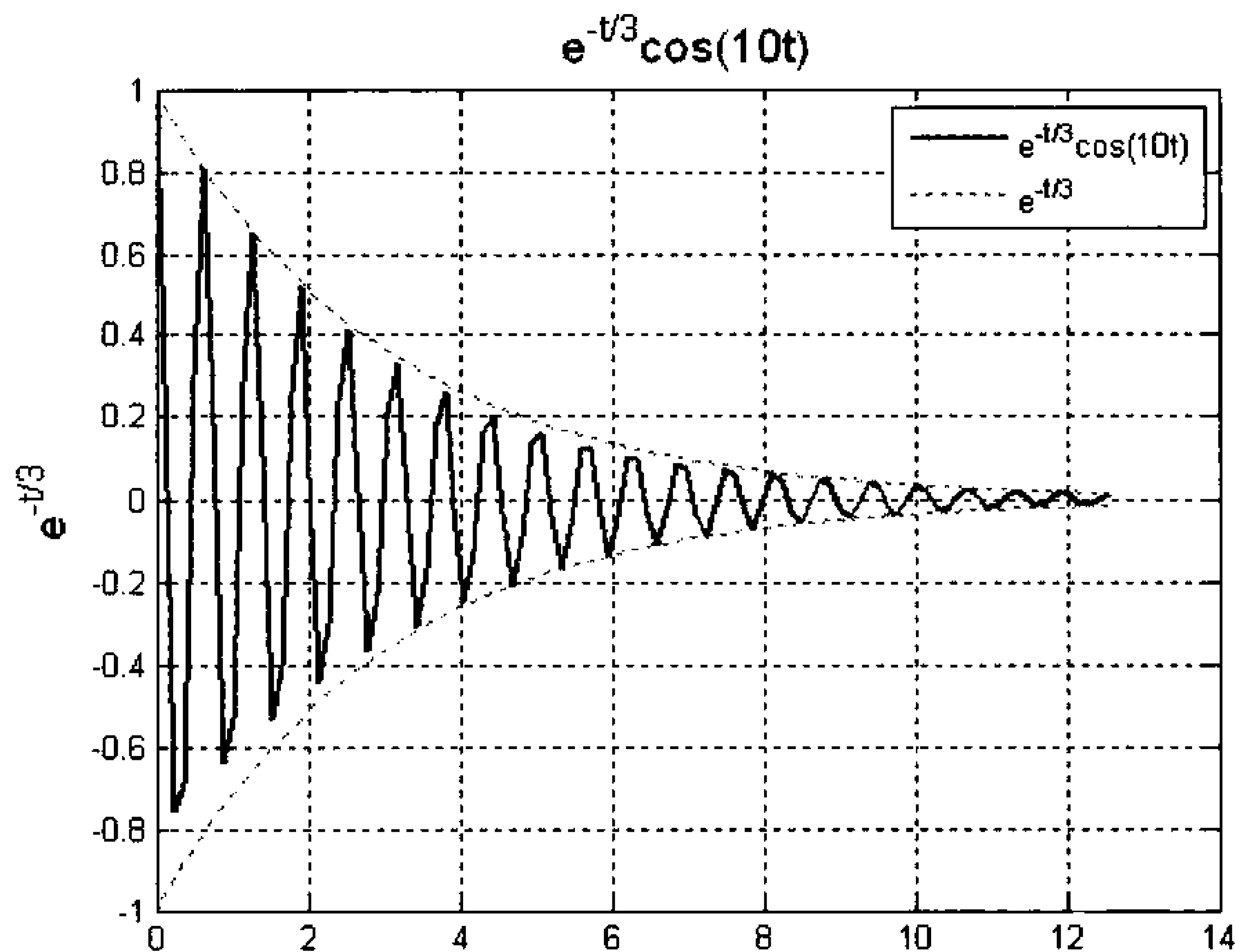


图 4-2　绘制曲线及其包络线

其他几个函数的用法与 plot 相似，在此不再一一介绍。

4.1.2 特殊二维绘图

在各种专业上常常碰到一些场合，需要把数据以分类的形式显示出来，例如，按月份组织年度销售收入、在信号处理中需要绘制时间信号的波形、气象工作者需要显示若干地区的平均气温数据等。为了满足这些特殊要求而需要采用特殊的平面图形。实际工作中人们习惯用直方图、条形图、扇形图等表达这些数据，为此 MATLAB 设计了一些专门用于绘制这些特殊平面图形的函数，使得这些工作变得非常简单。

1）条形图

在 MATLAB R2007 中，绘制数据的条形图使用函数 bar 或 barh，bar 函数绘制垂直的条形图，barh 函数绘制水平方向的条形图。bar 或 barh 函数输入的参数是向量或矩阵，如果输入的是向量则绘制每一个分量的条形图，如果输入的是矩阵则先对矩阵的每一行的分量条形进行分组，然后再分别绘制出来。这两个函数调用格式如表 4-3 所示。

表 4-3　bar、barh 函数调用格式

调 用 格 式	说　明
bar(y)、barh(y)	绘制 y 的条形图
bar(x,y)、barh(x,y)	在位置 x 上绘制 y 的条形图
bar(x,y,width)、barh(x,y,width)	同 bar(x,y)、barh(x,y)，但指定条形的相对宽度和一组内条形的间距
bar(…,'grouped')、barh(…,'grouped')	同 bar(x,y)、barh(x,y)，但指定为默认显示形式
bar(…,'stacked')、barh(…,'stacked')	绘制各行元素累加的条形图
bar(…,LineSpec)、barh(…,LineSpec)	同 bar(x,y)、barh(x,y)，但用指定的线型绘制条形图
h=bar(...)、h=barh(...)	返回绘制条形图的句柄属性值向量

例程 4.3 为几个条形图绘制示例。

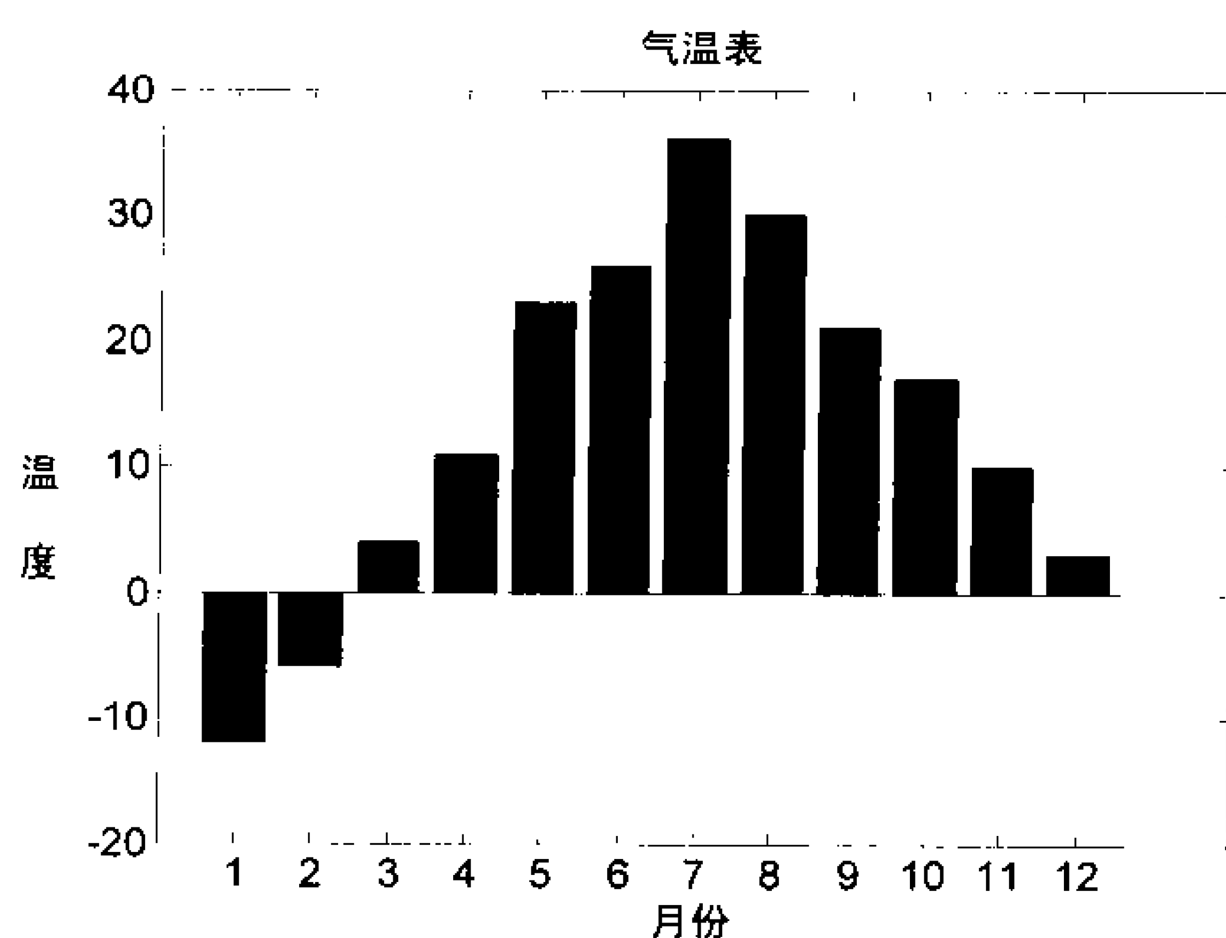
例程 4.3　条形图绘制示例

```
%假想某城市一年 12 月份平均气温数据，画出其条形图
>> x=1:12;
>> y=[-12,-6,4,11,23,26,36,30,21,17,10,3];
>> bar(x,y)
>> xlabel('月份'),ylabel('温度');
>> title('气温表')
```

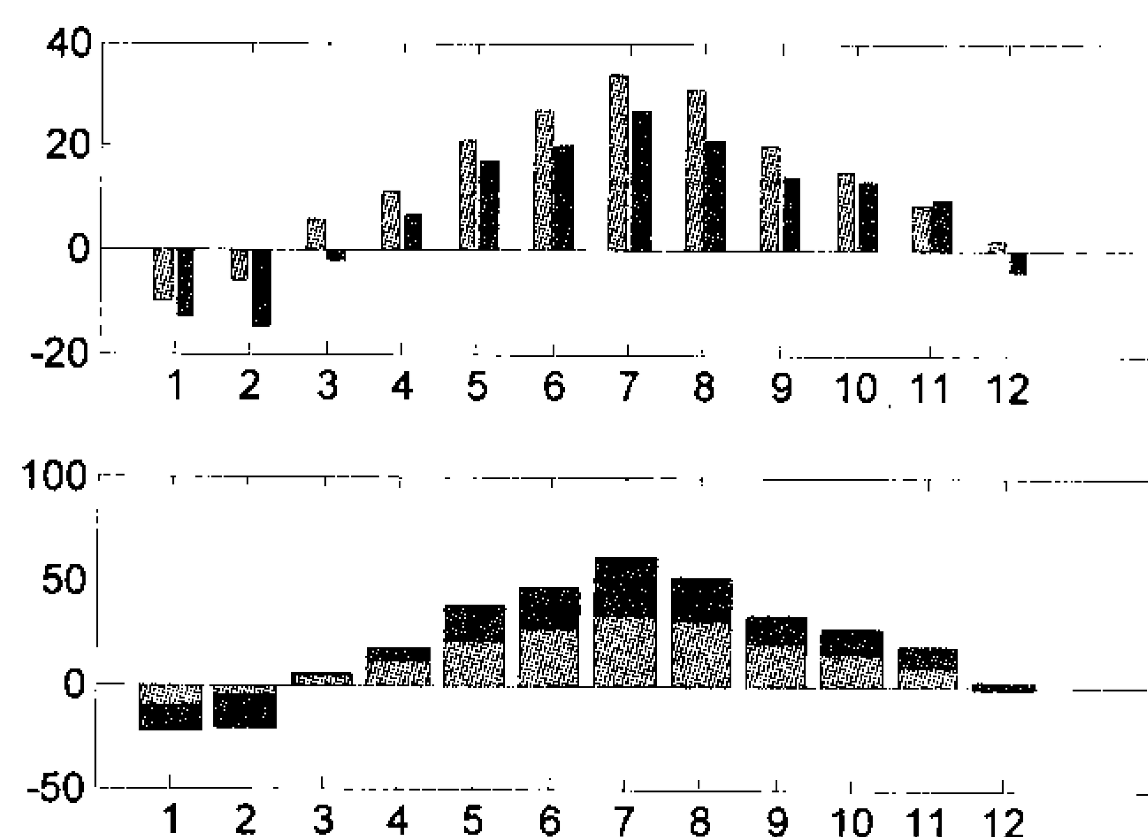
输入结果如图 4-3（a）所示。

```
%现在加上夜平均气温数据，再画出其条形图
>> figure(2)
>> x=1:12;
>> y=[-10,-6,6,11,21,27,34,31,20,15,9,2;-13,-15,-2,7,17,20,27,21,14,13,10,-4];
>> y=y';
>> colormap(cool)
>> subplot(2,1,1)
>> bar(x,y,'grouped')
>> subplot(2,1,2)
>> bar(x,y,'stacked')
```

输入结果如图 4-3（b）所示。



（a）平均气温条形图



（b）两种形式的条形图比较

图 4-3

这里的 y 必须转置成 12 行 2 列，它的行数必须等于向量 x 的长度。命令 colormap 用来把当前颜色映像的色调改为“cool”。我们分别画出了“grouped”和“stacked”两种形式的条形图。读者也可以自己动手试试三维条形图的绘制，编写过程和二维条形图相似。

在实际工程中，经常会用到误差条形图。误差条形图由函数 errorbar 生成，调用格式与 bar 函数相同。它可以沿着一条曲线绘制出其误差范围图。

例程 4.4 为绘制对应于正弦波信号的单位标准差对称误差值长条图。

例程 4.4 误差值长条图绘制示例

```
x=linspace(0,2*pi);y=sin(x);
e=std(y)*ones(size(x));%单位标准差
errorbar(x,y,e,'d');
```

得到的结果如图 4-4 所示。

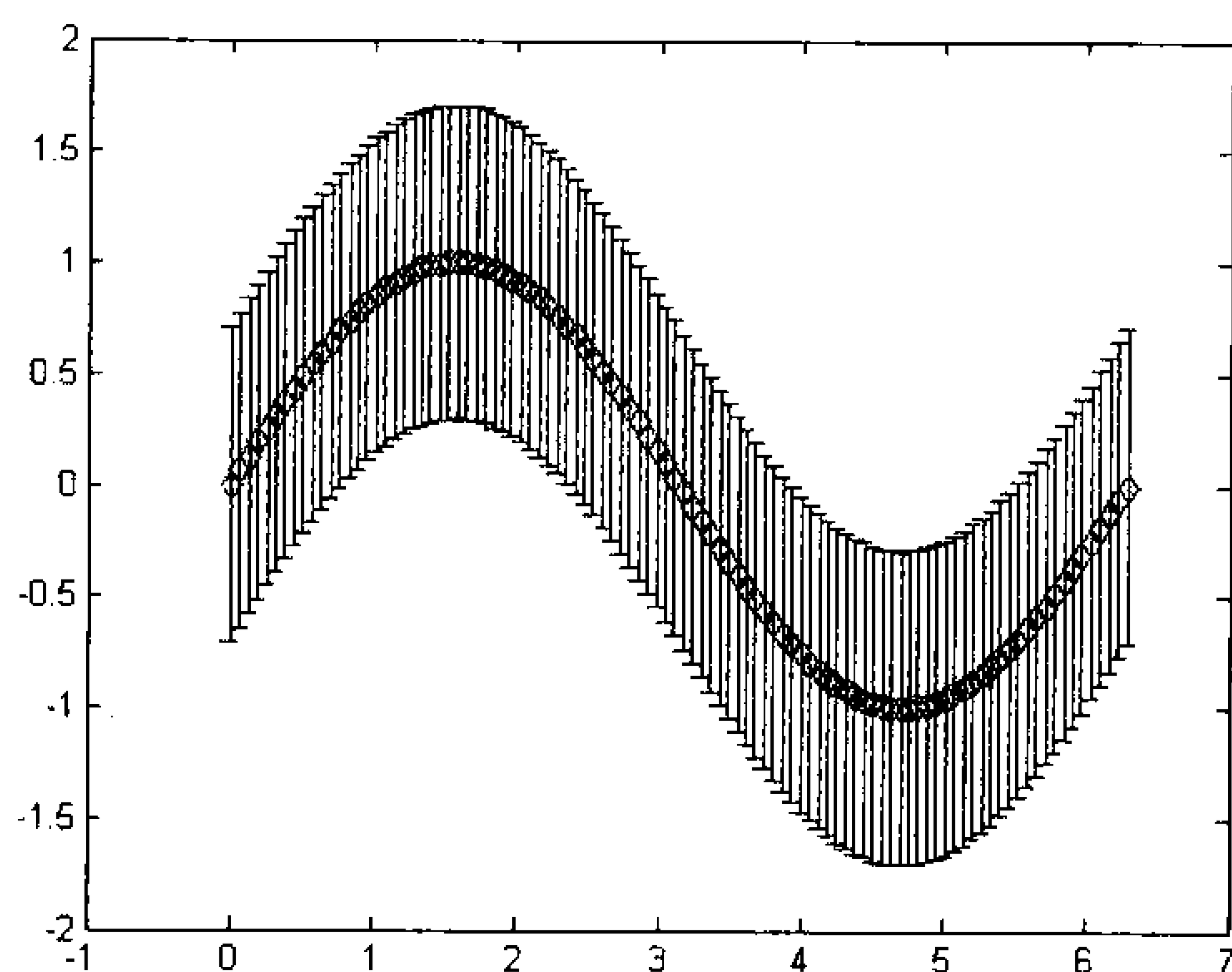


图 4-4 errorbar 绘图结果

2）面积图

绘制数据的面积图用 area 函数。area 函数根据向量或矩阵的列向量中的分量构成数据点，再将这些数据点连成一条或多条折线，然后用颜色填充折线下的面积，以此来显示一

个数值在该列所有数值总和中所占的比例。其调用格式如表4-4所示。

表 4-4 area 函数调用格式

调用格式	说明
area(y)	绘制向量 y 的面积图或矩阵 y 各列元素总和的面积图
area(x,y)	在 x 的位置上绘制 y 相应数据的面积图
area(…,ymin)	绘制面积图，但指定 y 方向上面积填充的最低限，y_{min} 默认值为 0
area(…,'ProName','ProVal',…)	绘制面积图，并为绘制的面积图设定属性和属性值
h=area(…)	返回绘制面积图的句柄属性值向量

例程4.5为使用函数 area 来绘制矩阵的面积图。

例程 4.5 面积图绘制示例

```
>> y=[3 2 -2 2 1;-1 3 3 7 2;-7 5 5 9 3];
>> area(y)
```

得到的结果如图4-5所示。

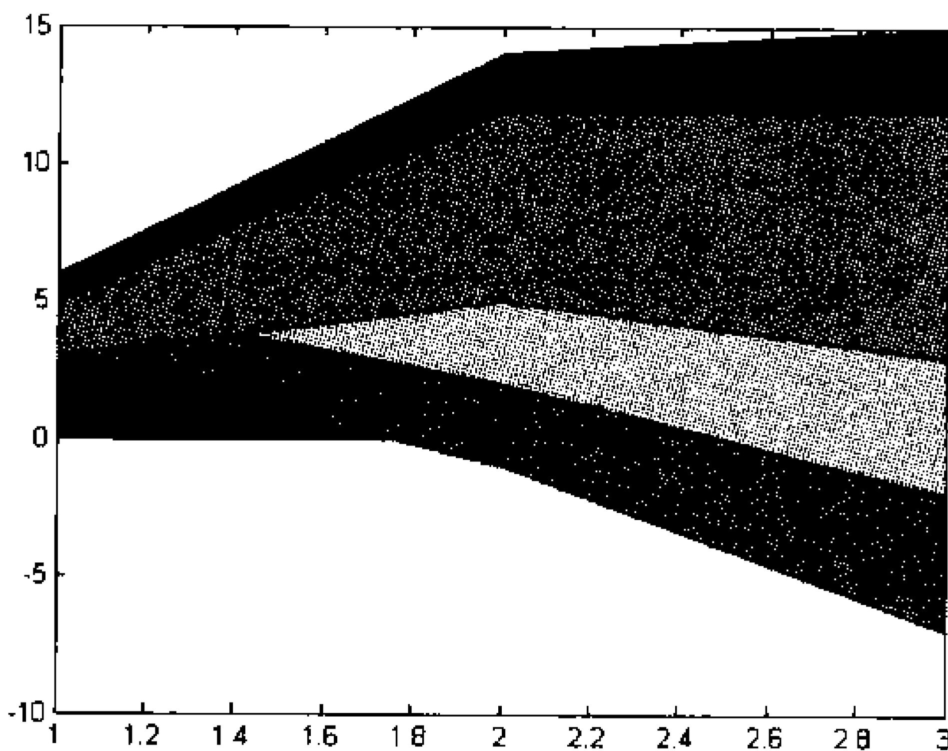


图 4-5 矩阵的面积图

3）柱形图

柱形图用来显示数据的分布情况，绘制二维柱形图使用 hist 函数，输入的参数 y 是向量或矩阵。如果是向量，则将向量中的元素按它们数值的范围分组，然后绘制柱形图；如果是矩阵，则将矩阵的每一列作为一个向量进行处理。该函数调用格式如表4-5所示。

表 4-5 hist 函数调用格式

调用格式	说明
hist(y)、n=hist(y)	绘制数据 y 的柱形图
hist (y,x)、n=hist(y,x)	同 hist(*y*)、*n=hist(y)*，但每一个柱形中心的位置放在 x 向量元素指定的位置
hist(y,m)、n=hist(y,m)	绘制数据 y 的柱形图，参数 m 指定柱形的个数
[n,X]=hist(…)	不绘制数据 y 的柱形图。只返回反映每个柱形中元素个数的向量和反映每个柱形频率的向量

例程4.6为 hist 函数应用示例。

例程 4.6 柱形图绘制示例

```
>> Y=randn(15000,2);hist(Y)
```

```
>> Y=randn(15000,1);hist(Y,30)
```

输出的结果如图 4-6 所示。

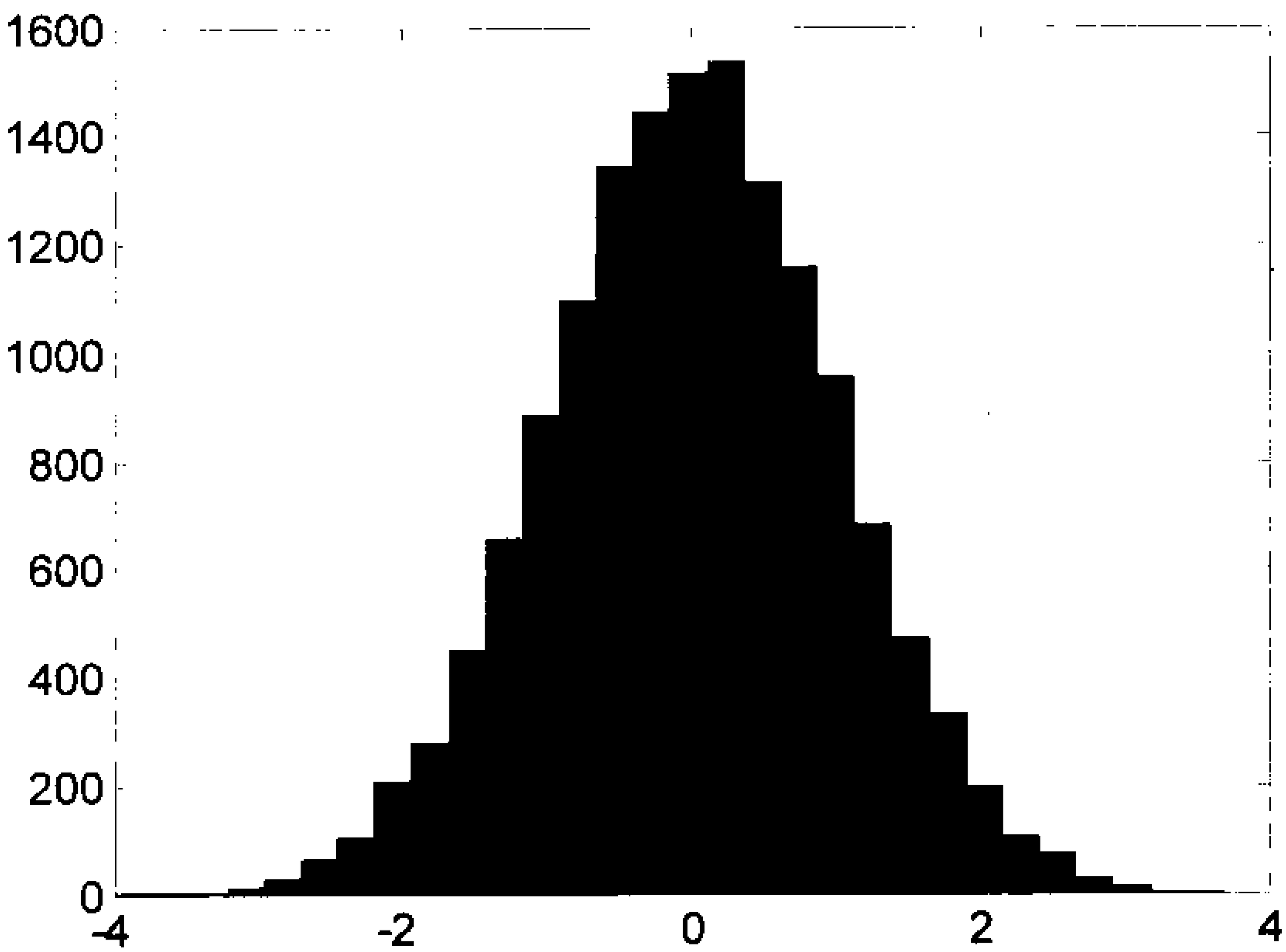


图 4-6 正态分布随机序列

4）火柴杆图

数据火柴杆图用于反映离散数据点离某一横轴的距离，图形是用垂直于横轴的线条和上端点处的小圆圈（默认点型）或其他典型表示，并在纵轴上标记数据值。绘制火柴杆图使用 stem 函数，输入的数据参数可以是向量也可以是矩阵。若数据为向量，则绘制向量每一个分量的火柴杆图；若数据为矩阵，则将矩阵分成行向量绘制每一分量的火柴杆图。stem 函数调用格式如表 4-6 所示。

表 4-6 stem 函数调用格式

调 用 格 式	说 明
stem(y)	绘制数据 y 的火柴杆图
stem(x,y)	在向量 x 指定的位置绘制 y 的火柴杆图
stem(…,'fill')	绘制数据的火柴杆图，参数'*fill*'默认值表示火柴杆顶端的小圆圈不填充颜色
stem(…,LineSpec)	以 *LineSpec* 确定的线型要素绘制数据的火柴杆图
h=stem(…)	返回绘制图形的句柄属性值向量

例程 4.7 为 stem 函数绘制火柴杆图示例。

例程 4.7 火柴杆图绘制示例

```
>> x=[1:12];
>> y=[342 200 87 912 1342 132 790 823 760 320 290 340];
>> stem(x,y);
>> title('年度销售收入')
>> xlabel('月份')
>> ylabel('销售额（单位：万欧元）')
```

输出的结果如图 4-7 所示。

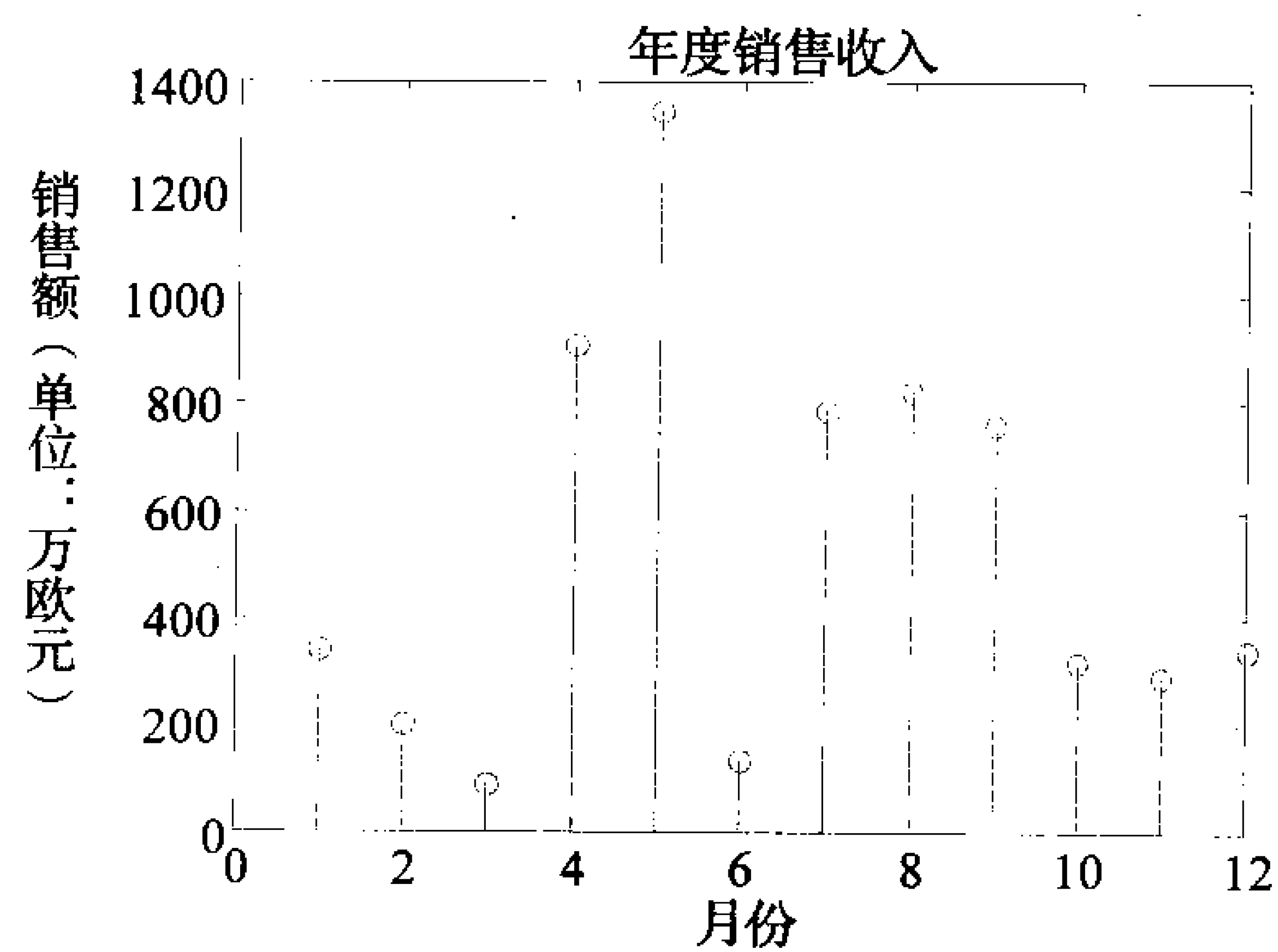


图 4-7　火柴杆图

5）扇形图

扇形图用于显示向量中的元素所占向量元素总和的百分比。MATLAB 中绘制扇形图使用 pie 函数，输入的参数为一个向量或矩阵。若输入的是向量，则绘制向量中各元素在所有元素之和中所占的比例；若输入的是矩阵，则绘制矩阵中各元素在矩阵所有元素和中所占的比例。pie 函数调用格式如表 4-7 所示。

表 4-7　pie 函数调用格式

调用格式	说　明
pie(x)	绘制数据 *x* 的扇形图
pie (x,explode)	同 pie(x)，参数 *explode* 是一个与 *x* 同样大小的向量

例程 4.8　使用函数 pie 绘制扇形图示例

```
>>x = [1 3 0.5 2.5 2];
>>explode = [0 1 0 0 0];
>>pie(x,explode)
>>colormap jet
```

得到的结果如图 4-8 所示。

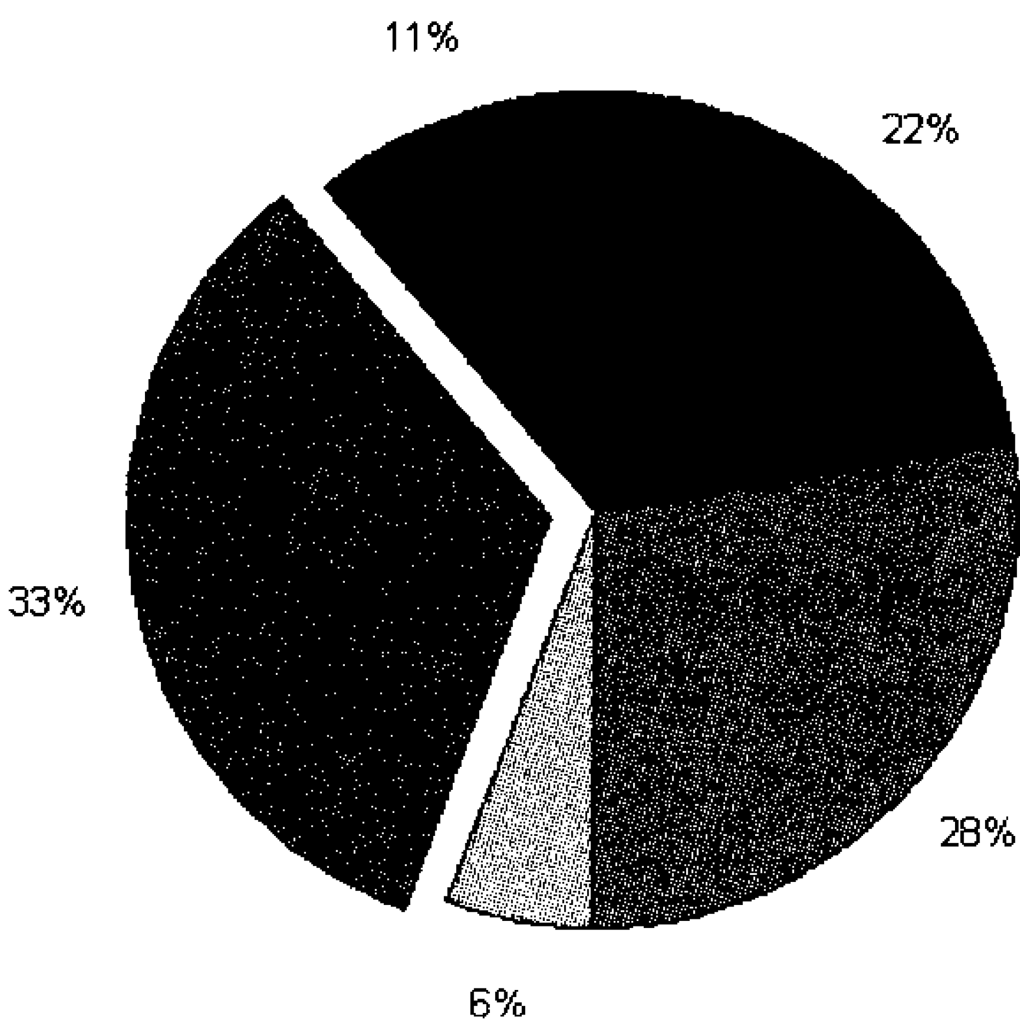


图 4-8　扇形图

6）羽毛图

如果用户需要产生沿着水平轴向外扩张的箭头，就可以使用 feather 产生类似羽毛的图形。羽毛图在横坐标上等距地显示向量，因此用户必须表示各个向量元素相对于原点的向量。feather 函数调用格式如表 4-8 所示。

表 4-8　feather 函数调用格式

调 用 格 式	说　　明
feather(u,v)	绘制由数据参数 *u*,*v* 确定的羽毛图。其中 *u* 是角度数据向量，*v* 是矢径数据向量
feather(z)	绘制复数向量 *z* 的羽毛图
feather(…,LineSpec)	以 *LineSpec* 确定的线型要素绘制数据的羽毛图

例程 4.9 为绘制-90°～90°的羽毛图。

例程 4.9　羽毛图绘制示例

```
theta=(-90:10:90)*pi/180;%将-90°~90° 间的角度转换为弧度
r=2*ones(size(theta));%定义极坐标的半径
%坐标转换，将极坐标系转换为笛卡儿坐标系
[u,v]=pol2cart(theta,r);
feather(u,v);
axis equal%产生 x 与 y 等长的坐标轴
```

得到的结果如图 4-9 所示。

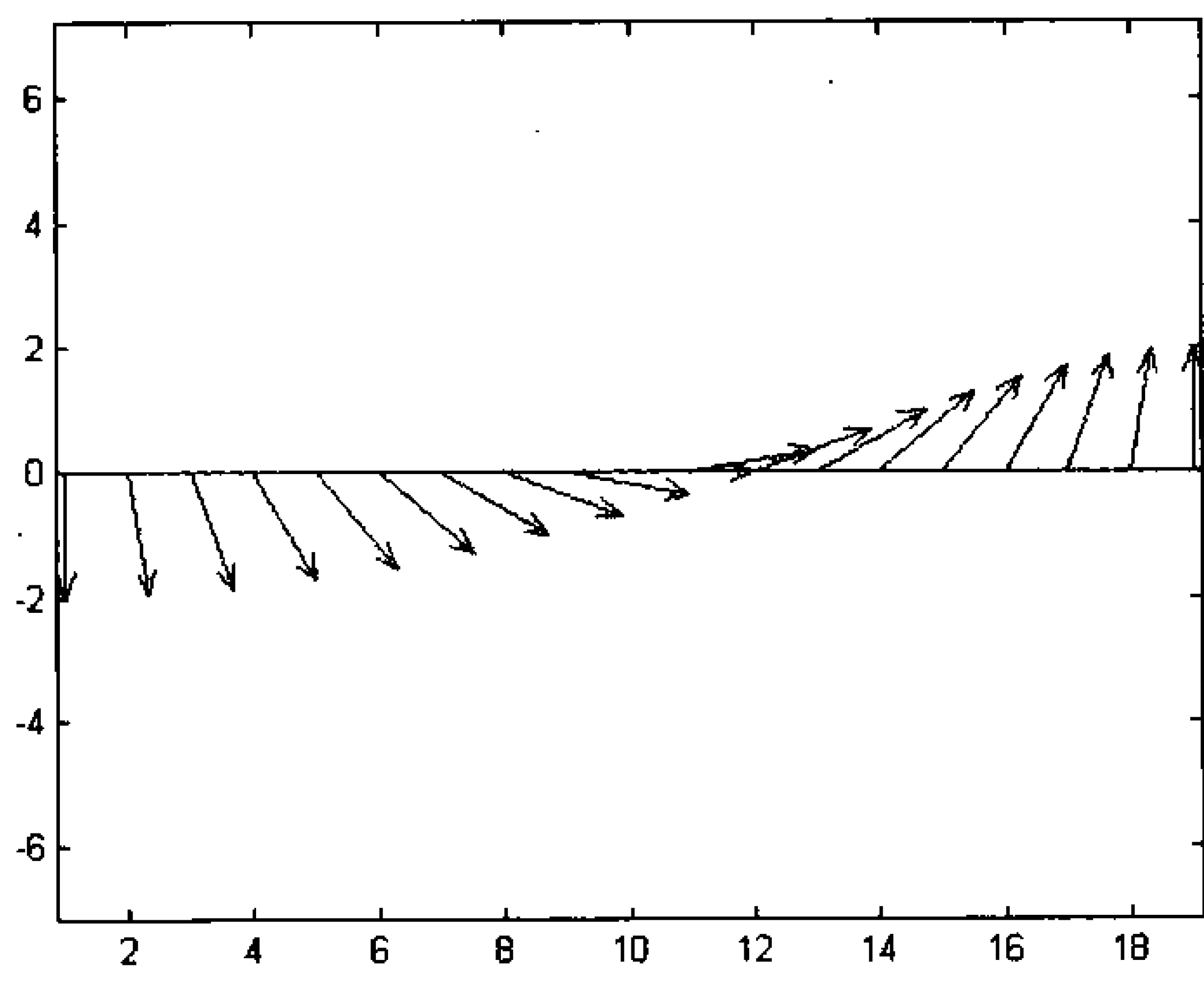


图 4-9　feather 绘图结果

7）其他二维绘图函数

MATLAB R2007 中除了以上绘图函数以外还有大量的绘图函数，如在同一窗口绘制不同坐标标度的多轴图形绘制函数 plotyy、向量图绘制函数 quiver、彗星图绘制函数 comet、等值线的绘图函数 contour、阶梯图绘制函数 stairs、玫瑰花绘制函数 rose，以及罗盘绘制函数 compass 等。

下面是关于这些函数的一些使用示例。

例程 4.10 按多轴标度图形的绘制方法在同一窗口绘制 $y=50e^{-\frac{x}{20}}\sin x$ 和 $y=\frac{1}{2}e^{-\frac{x}{2}}\cos x$ 在[0,30]上的图形。

例程 4.10　多轴标度绘图示例

```
>> x=[0:0.01:30];
>> y1=50*exp(-0.05*x).*sin(x);
>> y2=0.5*exp(-0.5*x).*cos(x);
>> plotyy(x,y1,x,y2,'plot')
```

得到的结果如图 4-10 所示。

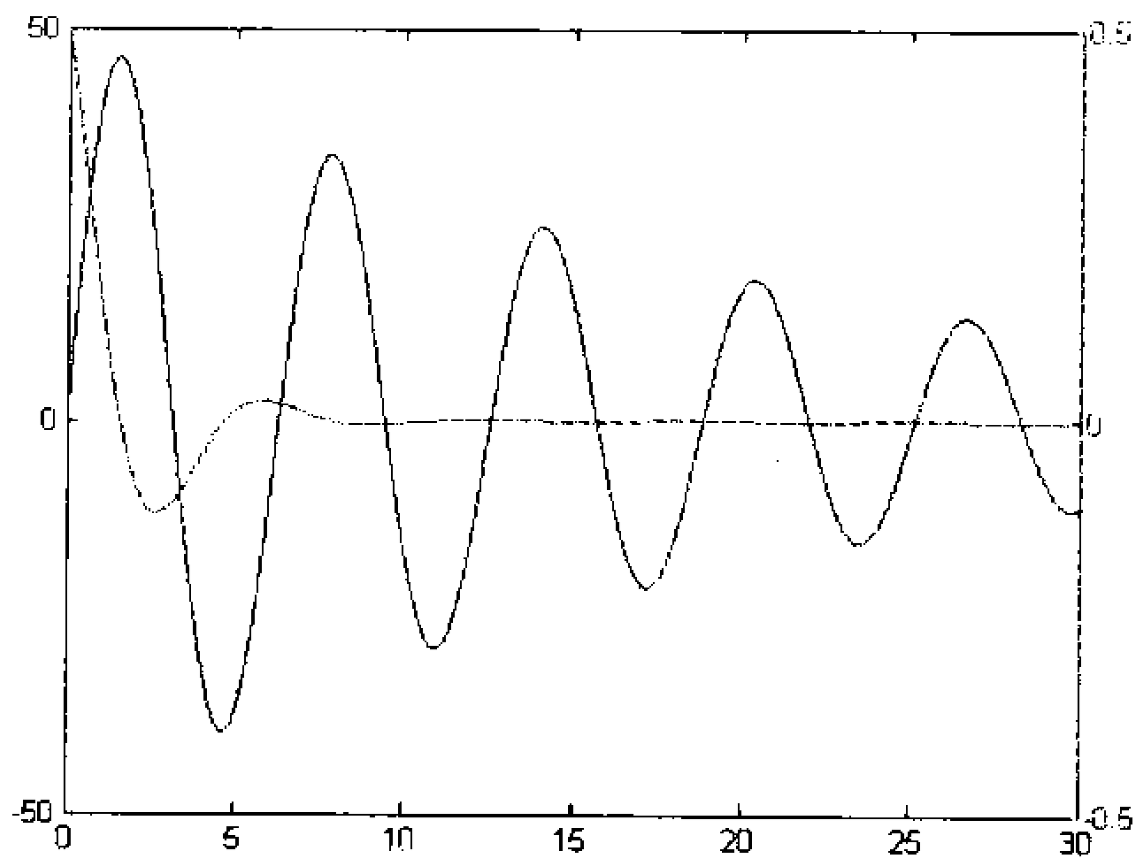


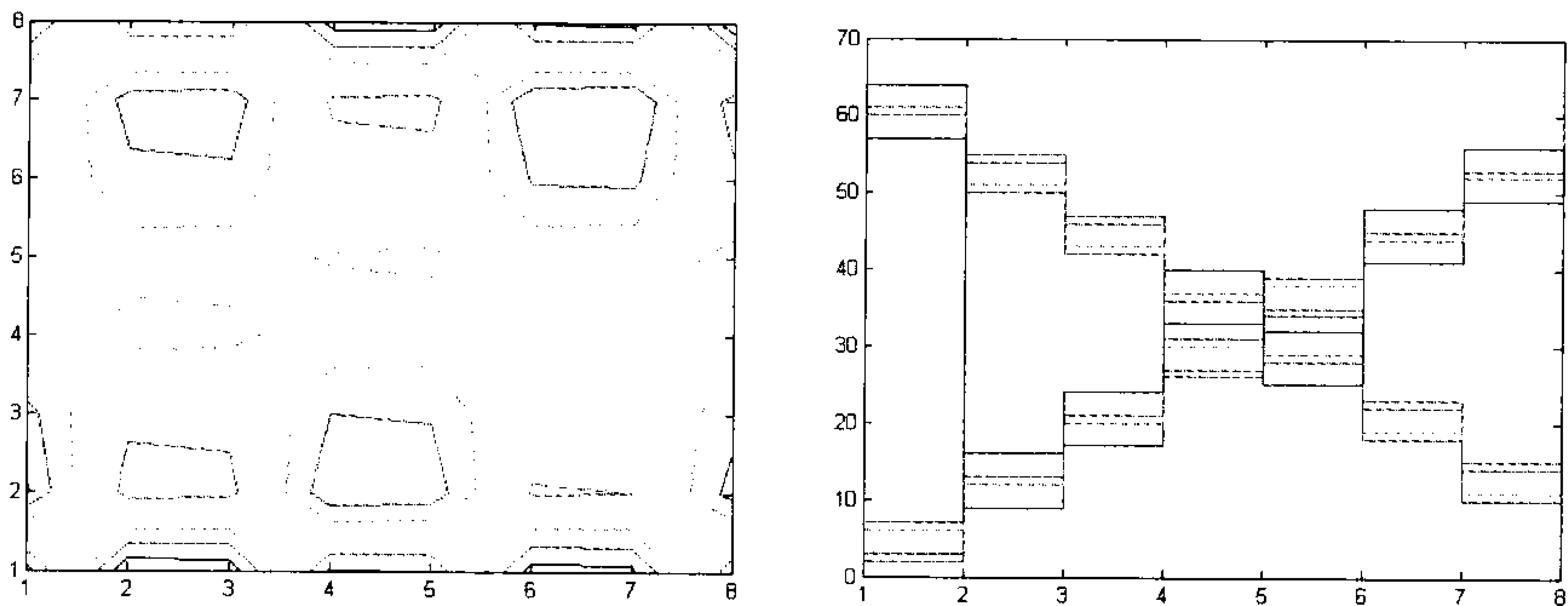
图 4-10　多轴标度图形

例程 4.11 为绘制 8 阶魔术矩阵的等值线图和阶梯图。

例程 4.11　等值线图和阶梯图绘制示例

```
>> A=magic(8);
>> contour(A);
>> stairs(A);
```

得到的结果如图 4-11 所示。



（a）8 阶魔术矩阵的等值线图　　（b）8 阶魔术矩阵的阶梯图

图 4-11

例程 4.12 为由 MATLAB 自行确定数据，绘制其玫瑰花图。

例程 4.12　玫瑰花图绘制示例

```
>> theta=rand(1,200)*2*pi;   %生成随机数据向量
>> rose(theta,25)            %绘制玫瑰花图
```

得到的结果如图 4-12 所示。

例程 4.13 为由 MATLAB 自行确定数据，绘制其罗盘图。

例程 4.13　罗盘图绘制示例

```
>> x=rand(20,1);
>> y=randn(20,1);
>> compass(x,y)
```

得到的结果如图 4-13 所示。

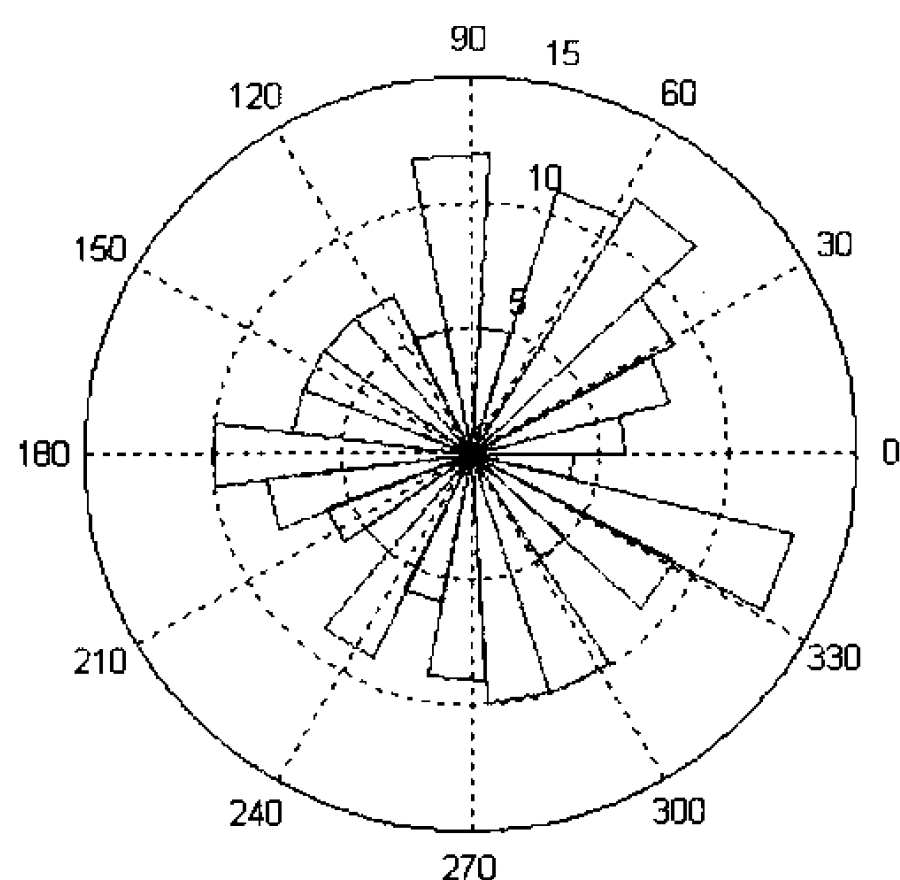


图 4-12　玫瑰花图

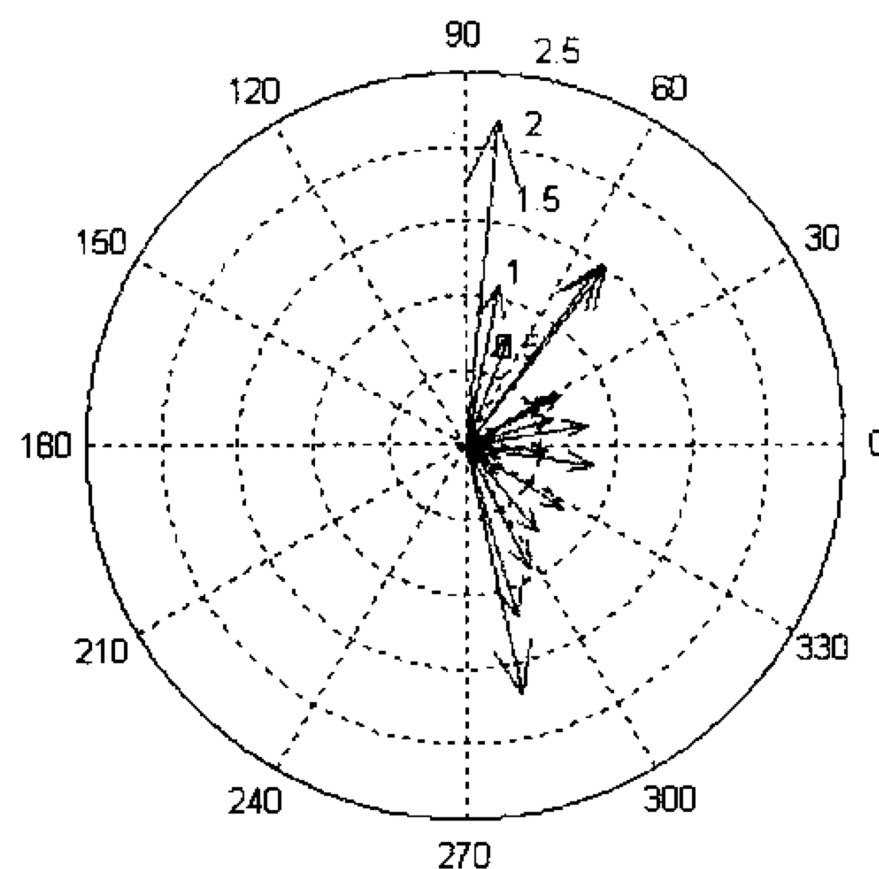


图 4-13　罗盘图

例程 4.14 绘制函数 $z=3xy\mathrm{e}^{-x^2-y^2}-1$ 的梯度向量图。

例程 4.14　梯度向量图绘制示例

```
>> [x,y]=meshgrid([-2:0.1:2]);      %建立栅格点数据向量
>> z=3.*x.*y*exp(-x.^2-y.^2)-1;     %计算函数值向量
>> [u,v]=gradient(z,0.2,0.2);       %计算梯度场向量
>> quiver(x,y,u,v,2)                %绘制梯度场向量图
```

得到的结果如图 4-14 所示。

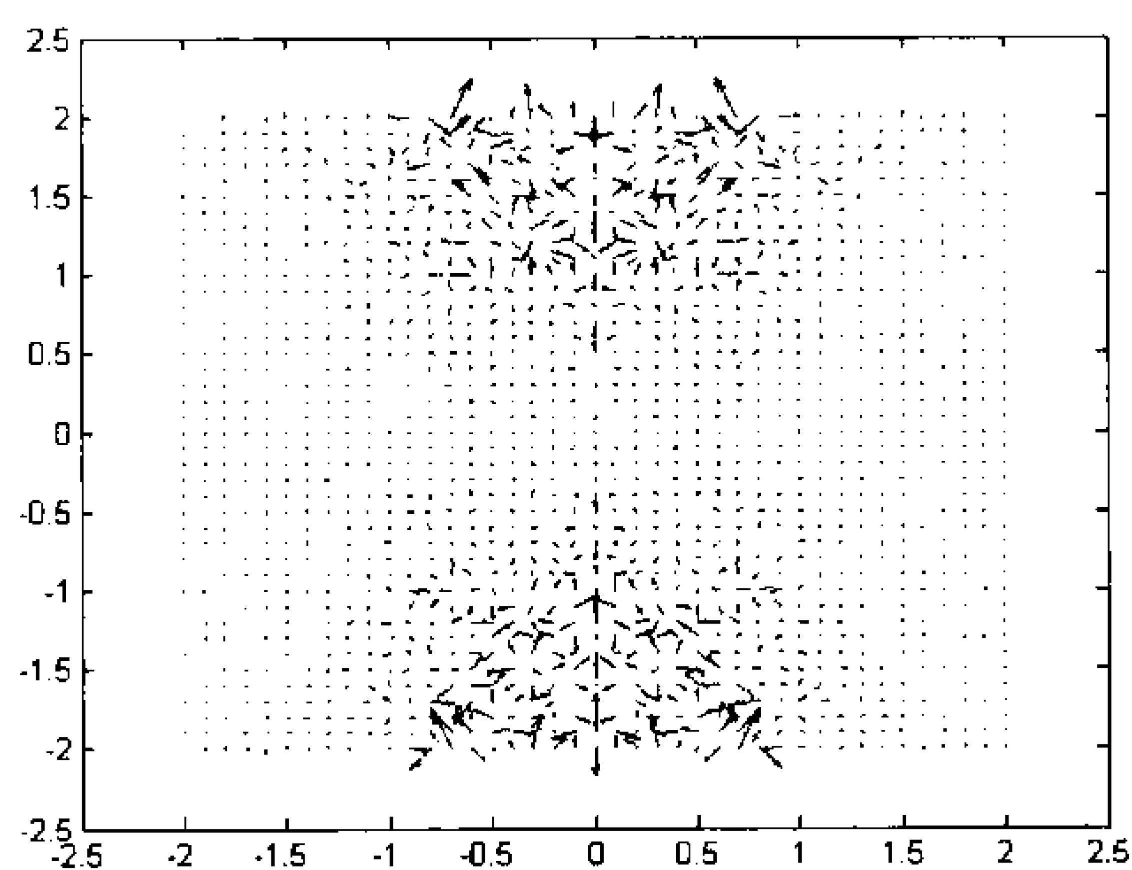


图 4-14　函数的梯度场向量图

这里仅列举几个比较典型且常用的函数的示例，限于篇幅，其他的就不再一一说明，请读者参考其他资料。

4.1.3　二维绘图的进阶功能

在绘图过程中，若想获得图面上某一点的坐标值，除了在 GUI 中通过 CurrentPoint 属性外，

还可以利用函数 ginput，很方便地查询当前窗口中特定点的坐标值。另外，若要对图面进行缩放操作，除了可以通过绘图窗口的工具栏外，还可以通过以下介绍的 zoom 函数来完成。

1）zoom 函数

函数 zoom 用来指定是否可以对图形进行缩放，这在分析数据量比较大的图形时经常用到，它的调用方法如表 4-9 所示。

表 4-9　zoom 函数调用格式

调用格式	说　明
zoom on	交互式图形放大与绘图原图大小功能。执行此函数后，可以使用鼠标去选取欲放大（按住左键拖曳）的区域，或是直接在该区域上单击鼠标左键即可产生放大效果，若双击鼠标左键则恢复原图大小
zoom off	停止放大缩小
zoom out	恢复为原图大小
zoom reset	系统将记住当前图形的放大状态，作为后续放大状态的设置值。因此以后使用 zoom out 时，图形并不会恢复为原图大小，而是返回 reset 时的放大状态的大小
zoom	用于切换放大的状态：on 和 off
zoom xon 、zoom yon	仅对 *X* 轴或 *Y* 轴进行放大
zoom(factor)	用 zoom 系数 *factor* 进行放大或缩小，不影响交互式（zoom on）放大的状态。若 *factor*>1，系统将图形放大 *factor* 倍；若 0<*factor*≤1,系统将图形放大 1/*factor* 倍
zoom(fig, option)	指定对句柄值为 fig 的绘图窗口的二维图形进行放大，其中参数 option 为 *on*、*off*、*xon*、*yon*、*reset*、*factor* 等
h = zoom(figure_handle)	返回操作的句柄属性值向量

2）ginput 函数

函数 ginput 允许鼠标获取图形上坐标轴范围内点的坐标，它有如下 3 种调用方式。

表 4-10　ginput 函数调用格式

调用格式	说　明
[x,y]=ginput(n)	从图形中获得 *n* 个点的坐标值，获得的数据保存在长度为 *n* 的向量 *x* 和 *y* 中
[x,y]=ginput	从图形中获得任意多个点的坐标，直到按下回车键为止
[x,y,button]=ginput(n)	返回值中增加了 *button* 向量，该向量中的元素为整数，反映选取数据点时按下了哪个鼠标键（左、中、右键分别对应 1、2、3）或返回使用的键盘上的键的 ASCII 码值

调用 ginput 函数后，在窗口中鼠标箭头会变成十字形的光标，移动鼠标，光标随之移动，在关心的数据点上单击鼠标左键，该点的坐标就被记录下来，直到点数达到指定的个数或按下回车键终止取值为止。

4.1.4　线型、顶点标记和颜色

在用户没有指定的情况下，MATLAB 的绘图函数会默认地选择实线线型，并以一个默认的颜色顺序绘制每一个图形的颜色。不过，MATLAB 的绘图函数是允许用户指定图形的线型和颜色的，另外，用户还可以指定用于区别数据点的标记。要实现这些定制化操作，用户只需将表 4-11 中的符号以字符串的形式传递给 MATLAB 绘图函数就可以了。

表 4-11 MATLAB 颜色、标记和线型允许设置值

符 号	颜 色	符 号	标 记	符 号	线 型
b	蓝色	.	点号	—	实线
g	绿色	。	圆圈	:	点线
r	红色	×	叉号	-.	点画线
c	青色	+	加号	--	虚线
m	品红	*	星号		
y	黄色	s	方形		
k	黑色	d	菱形		
w	白色	∨	向上三角形		
		∧	向下三角形		
		<	向左三角形		
		>	向右三角形		
		p	五角形		
		h	六角形		

如果用户没有声明颜色并且在使用默认颜色机制，MATLAB 就为每一条新增加的线从表 4-11 的前 7 种颜色的蓝色和圆圈开始从上到下依次选择颜色和标记类型。默认的线型是实线，除非用户显式地声明了另一种线型。没有默认的标记。如果没有选择标记，那么就不会画出标记。

如果字符串中包含了颜色、标记和线型，那么就将颜色应用到标记和线条中，为了标记声明一种不同的颜色，可用不同的声明字符串再绘制一次相同的数据，如例程 4.15。

例程 4.15 线型和颜色设置示例

```
>> x=linspace(0,2*pi,30);
>> y=sin(x);
>> z=cos(x);
>> plot(x,y,'b:p',x,z,'c-',x,1.2*z,'m+')
```

程序运行结果如图 4-15 所示。

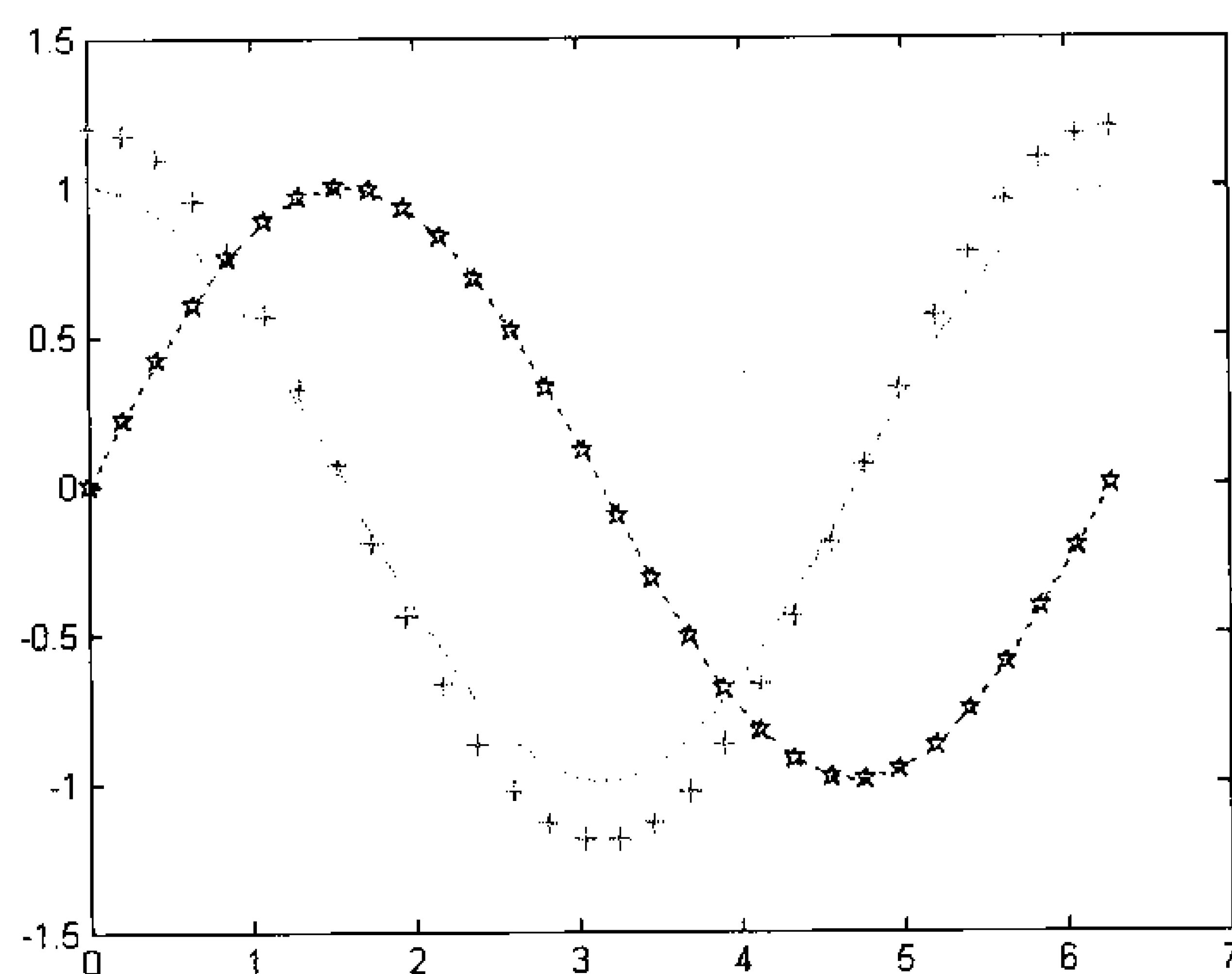


图 4-15 线型和标记

例程 4.15 在 0~2*pi 的范围内生成了 30 个数据点作为图形的水平坐标轴，然后生成了一个向量 y 包含与 x 相对应的正弦值，一个向量 z 包含与 x 相对应的余弦值。

4.1.5 分格线控制和图形标注

分格线和图形标注都是为了帮助图形阅读者更好地理解图形含义，同时对图形反映的数据信息能更好地把握。

1. 分格线控制

MATLAB 的默认设置是不画分格线，它的疏密取决于坐标刻度，如果想改变分格线的疏密，必须先定义坐标刻度，其调用格式及意义如表 4-12 所示。

表 4-12 分格线控制函数用法及意义

函数用法	描 述
grid	是否画分格线的双向切换指令（使当前分格线状态翻转）
grid on	画出分格线
grid off	不画分格线
box	坐标形式在封闭式和开启式之间切换指令
box on	使当前坐标呈现封闭形式
box off	使得当前坐标呈现开启形式



默认情况下，所画坐标呈封闭形式。

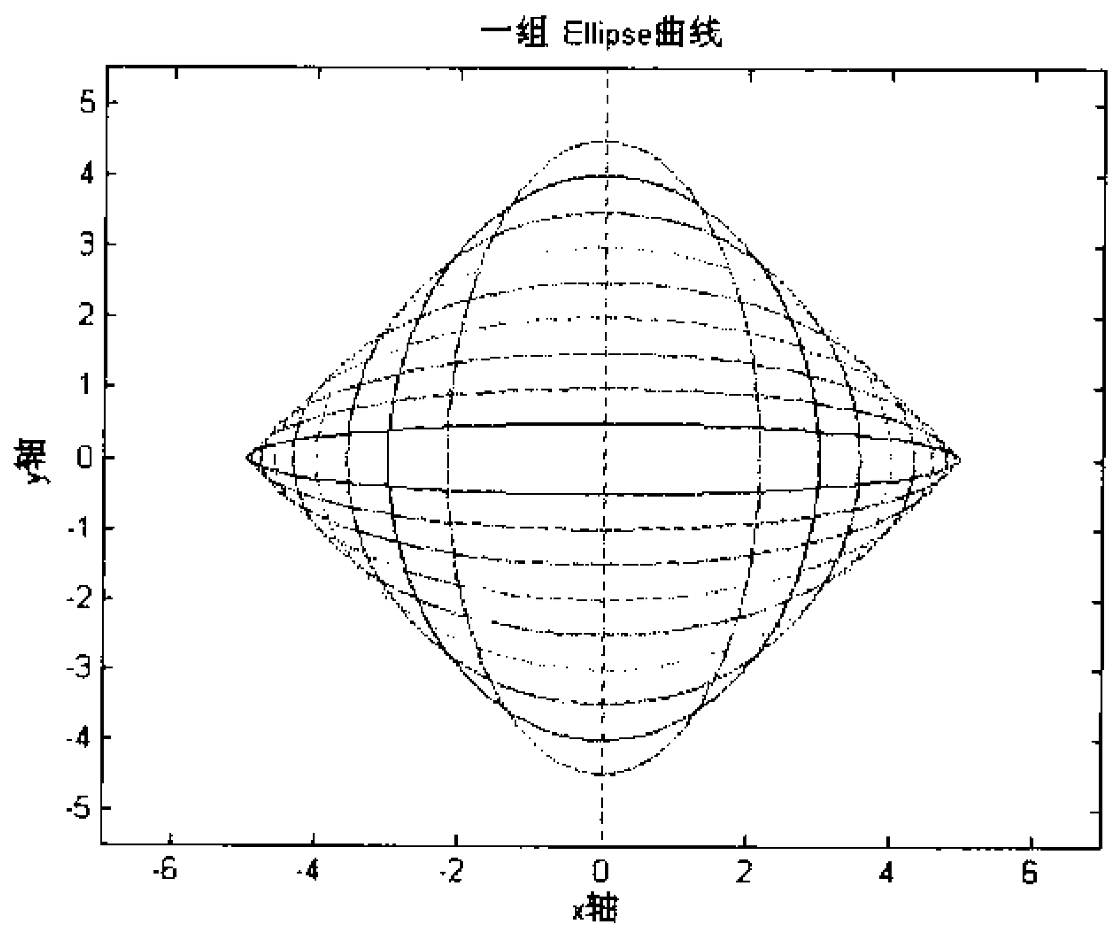
例程 4.16 分格线控制示例

```
>>th=[0:pi/50:3*pi]';
>>a=[0.5:0.5:5.5];Y=cos(th)*a;X=sin(th)*sqrt(25-a.^2);
>>plot(X,Y)
>>axis('equal');
>>xlabel('x 轴'),ylabel('y 轴')
>>title('一组 Ellipse 曲线')
```

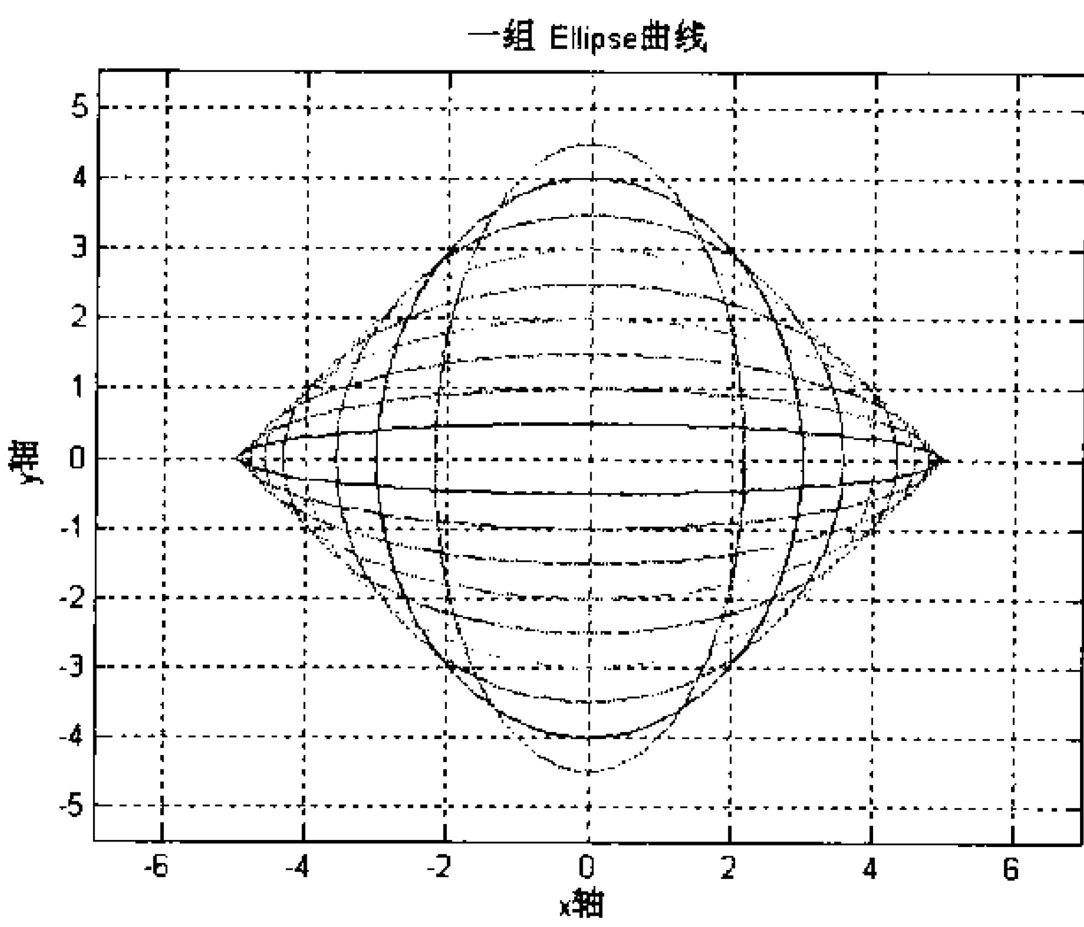
图 4-16(a)为运行结果，现在对结果图添加分格线。

```
>>grid on,box on;
```

得到的结果如图 4-16(b)所示。



（a）分格前



（b）分格后

图 4-16 一组椭圆图

2．图形标注

调用函数对图形进行标志的方法如下所示。

1）坐标轴名 label

给坐标轴 x，y，z 加标注，只要调用相应的函数 xlabel、ylabel 和 zlabel 即可。以函数 xlabel 为例，其调用格式如下：

```
xlabel('text','ProName1', 'ProVal1',' ProName2', 'Proval2',...)
```

其中“text”为要添加的标注文本，“ProName”指该文本的属性，“ProVal”为相应的属性值。该命令把文本“text”按照设置的格式添加到 x 轴的下方。

2）书写图名 title

给图形加标题的是函数 title，它的调用格式和 xlabel 类似，如下：

```
title('text', 'ProName1', 'ProVal1',' ProName2', 'Proval2'...)
```

区别只是 title 函数把文本“text”加到了图形的上方。

可以在函数 xlabel、ylabel 和 title 中设置的属性有很多，比如：字体粗细、字体角度和字号大小等，我们将在后面结合图形属性一并介绍。另外，在函数 title 的文本字符串中斜线“\”是引导特征字符串的标志符号。使用特征字符串可以把很多数学公式或工程中的 text 符号标注到图形上，大大增加了标注的灵活性。MATLAB 中的特征字符串及其对应的 text 符号可对应帮助指南或在线联机帮助进行查询。



title 命令要写到 plot 命令之后，否则不起作用；特征字符串是区分大小写的。

3）添加图例 legend

除了给图形加标题、标注和文本，MATLAB 还提供了函数 legend，它用文本确认每一个数据集，为图形添加图例便于图形的观察和分析。该函数的一般调用格式如下：

```
legend(string1,string2,string3,...)
```

只要指定标注字符串，该函数就会按顺序把字符串添加到相应的曲线线型符号之后。MATLAB 还可以很方便地对图例进行调整：用鼠标左键选中图例拖动，就可以移动图例到需要的位置，用鼠标左键双击图例中的某个字符串就可以对该字符串进行编辑。

在 legend 函数中加入一个参数也可以对图例的位置做出规定，格式如下：

```
legend(string1,string2,string3,...,Pos)
```

式中“Pos”是一个指定的位置，可以是一个 1×4 的位置向量或如表 4-13 所示的几个字符串参数。

表 4-13　字符串参数

Pos 功能字符串	描　述	Pos 功能字符串	描　述
'North'	在绘图框内接近顶部的位置	'NorthEast'	在绘图框内的右上方位置
'South'	在绘图框内的底部位置	'NorthWest'	在绘图框内的左上方位置
'East'	在绘图框内的右方位置	'SouthEast'	在绘图框内的右下方位置
'West'	在绘图框内的左方位置	'SouthWest'	在绘图框内的左下方位置
'NorthOutside'	在绘图框外接近顶部的位置	'WestOutside'	在绘图框外的左方位置
'SouthOutside'	在绘图框外的底部位置	'Best'	在绘图框内的抵触绘图资料

（续表）

Pos 功能字符串	描　述	Pos 功能字符串	描　述
'EastOutside'	在绘图框外的右方位置	'BestOutside'	在绘图框外不要使用空格
'NorthEastOutside'	在绘图框外的右上方位置	'SouthEastOutside'	在绘图框外的右下方位置
'NorthWestOutside'	在绘图框外的左上方位置	'SouthWestOutside'	在绘图框外的左下方位置

例程 4.17 为上述几种标注方法示例。

例程 4.17　上述几种标注方法示例

```
>> t=0:0.2:2*pi;
>> plot(t,sin(t),'>',t,cos(t),'+');
>> xlabel('x'),ylabel('y');
>> title('sin(x) and cos(x)');
>> legend('sin wave','cos wave')
```

输出结果如图 4-17 所示。

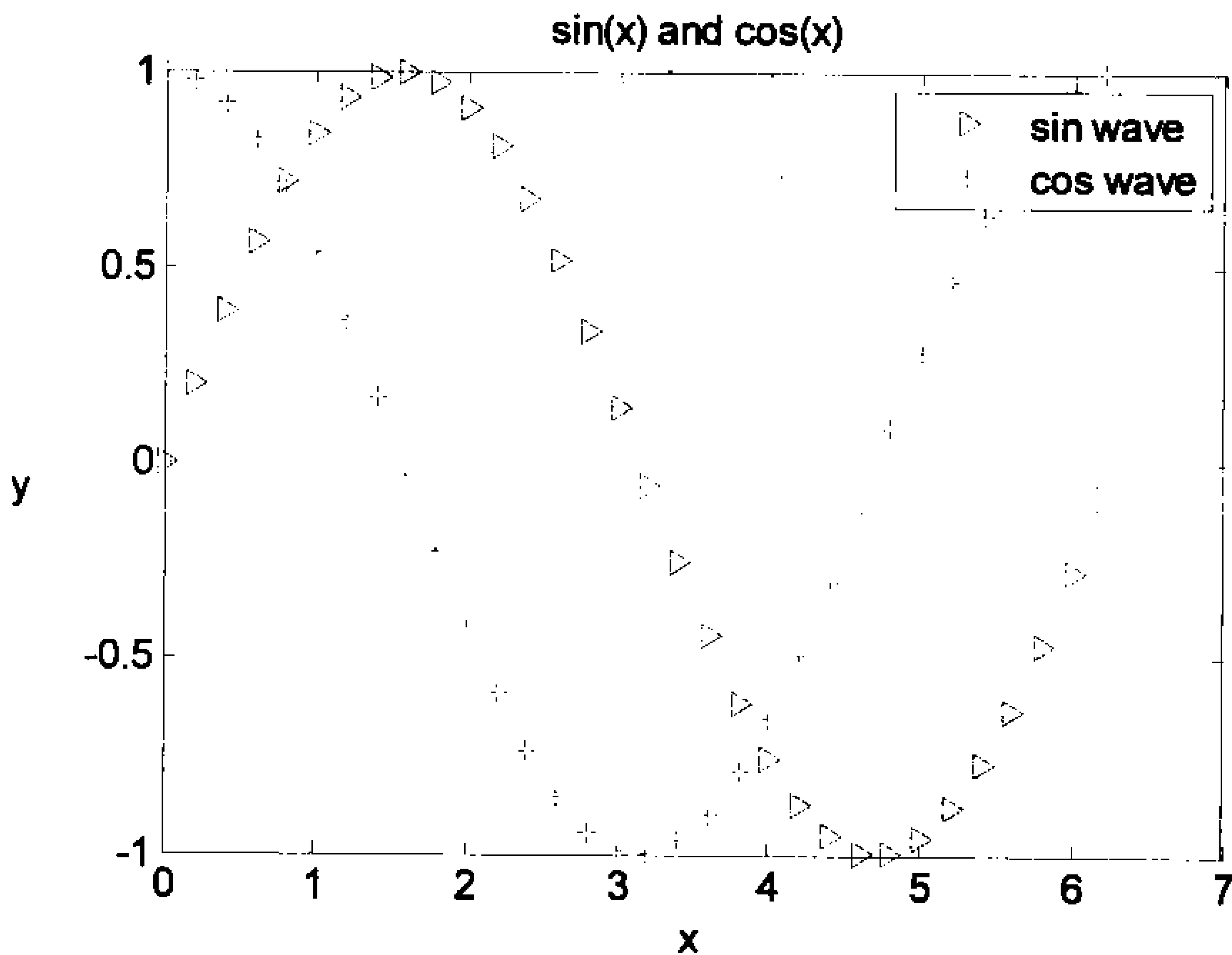


图 4-17　添加标注示意图

3．添加文本字符串 text

除了在坐标轴上能够作标志以外，MATLAB 还可以用 text 函数在图形窗口的任意位置加入文本字符串。该函数调用格式如表 4-14 所示。

表 4-14　text 函数调用格式

调 用 格 式	说　明
text(x,y,'string')	在指定的点（*x*,*y*）添加字符串 *string*
text(x,y,z,'string')	在三维空间上添加字符串
text(x,y,z,'string','ProName','ProVal'....)	在指定的位置和设定属性值时添加字符串到“当前”轴对象上
text('ProName','ProVal',....)	完全忽略轴坐标，定义属性对
h = text(...)	返回句柄属性值向量

例程 4.18 为 text 函数应用示例。

例程 4.18　运用 text 函数添加标注示例

```
>> clear;
>> x=0:pi/20:2*pi;
>> plot(x,sin(x));
>> text(pi,sin(pi),'\leftarrowsin(x)=0');
>> text(3*pi/4,sin(3*pi/4),'\leftarrowsin(x)=0.707');
>> text(0,-0.7,['画图时间:',date])
```

输出结果如图 4-18 所示。

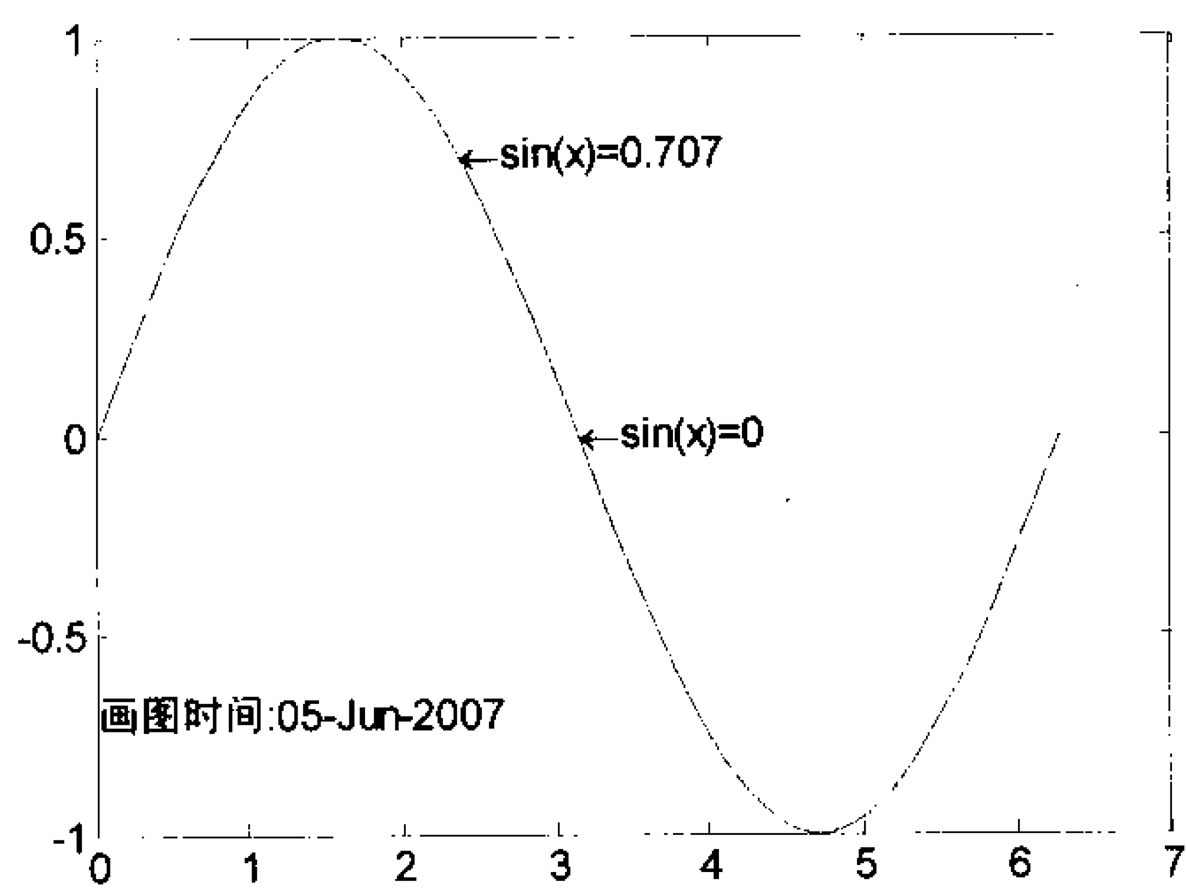


图 4-18　图形添加文本的输出图

text 的默认方式是从插入点的右边开始写文字；date 函数返回当前日期，而且返回值为字符串格式，不需要进行类型转换，可以直接和其他字符串连接。

4.1.6　屏幕刷新

由于屏幕刷新相对而言很消耗时间，因此 MATLAB 并不总是在每个图形命令之后都刷新屏幕。举一个简单的例子，例如，如果用户在 MATLAB 提示符下逐个输入下面的命令，MATLAB 就会在执行每个命令（包括 plot、axis 和 grid）时刷新屏幕。

```
>> x=linspace(0,2*pi);y=sin(x);
>> plot(x,y)
>> axis([0 2*pi -1.2 1.2])
>> grid
```

但如果用户将这些命令在同一行输入，例如：

```
>> plot(x,y),axis([0 2*pi -1.2 1.2]),grid
```

那么 MATLAB 只对屏幕进行一次刷新。另外，如果上述命令出现在一个 M 脚本文件或 M 函数文件中，MATLAB 也只进行一次屏幕刷新。

总的来说，在 MATLAB R2007a 中，以下 6 种情况可以导致屏幕刷新，如表 4-15 所示。

表 4-15　导致屏幕刷新的原因

编　号	名　称	编　号	名　称
1	在命令窗口中返回到下一个 MATLAB 提示符时，即用户在输入命令后按回车键	4	遇到一个临时中止执行的函数，比如 pause、keyboard、input 和 waitforbuttonpress
2	执行一个 getframe 命令	5	执行一个 drawnow 命令
3	执行一个 figure 命令	6	重新设置一个图形窗口的大小

在上述 6 种情况中，drawnow 命令可使用户在任何时候强制 MATLAB 刷新屏幕。

4.2 三维图形绘制

在实际工程应用中，需要对三维数据进行分析处理，如处理二维函数对应的曲线，这些问题比较复杂抽象。利用 MATLAB 绘制三维图形进行辅助分析，就可以很好地解决这类难题。因为 MATLAB 可以在三维空间准确地表示复杂的二维函数对应的曲线，并对图形进行各种处理。

为了显示三维图形，MATLAB 提供了丰富的三维绘图函数。有些函数能在三维空间中画线，而另一些可以画曲面和框架。另外，颜色可以代表第四维。这些函数的开发使用，让 MATLAB 具有了强大的三维图形处理功能，包括三维数据显示、空间曲线、曲面、分块、填充及视角变换、旋转、隐藏等功能和操作。本节将围绕这些问题展开讨论。

4.2.1 基本三维绘图

MATLAB 三维绘图函数如表 4-16 所示。

表 4-16 三维绘图函数

函数名称	用法	函数名称	用法
plot3	三维曲线图	fill3	三维填充多边形图
stem3	三维离散序列图	contour	等高线图
sphere	三维立体圆球图	cylinder	圆柱体图
slice	立体切片图	trimesh	三角网目图
quiver	向量图	mesh	网目图
surf	表面图	waterfall	瀑布图

1）三维曲线图

和二维情形下的 plot 函数相对应，在三维情况下，我们有函数 plot3，它将 plot 函数的特性扩展到了三维区间。两者之间的区别在于 plot3 增加了第三维数据，调用格式如表 4-17 所示。

表 4-17 plot3 函数调用格式

调用格式	说明
plot3(x,y,z)	以默认线型属性绘制三维点集(x_i,y_i,z_i)确定的曲线。x,y,z 为相同大小的向量或矩阵
plot3(x,y,z,s)	以参数 S 确定的线型属性绘制三维点集(x_i,y_i,z_i)确定的曲线。x,y,z 为相同大小的向量或矩阵
plot3(x1,y1,z1,S1,...)	绘制多个以参数 S_i 确定线型属性的三维点集(x_i,y_i,z_i)确定的曲线。x,y,z 为相同大小的向量或矩阵
plot3(...,'ProName','ProVal',...)	绘制三维曲线，根据指定的属性值设定曲线的属性
h=plot3(...)	返回绘制的曲线图的句柄属性值向量

上一节介绍的二维图形的所有基本特性都可以直接或经简单处理后应用在三维图形中。另外，利用命令 grid 也可以在三维图形中生成三维栅格，利用命令 box 则可以生成一个包围图形的三维边框（与 plot 一样，plot3 的默认设置也为 grid off 和 box off）。用户也可以利用命令 text(x,y,z,'string')将一个字符串'string'放置在由 x、y、z 指定的位置，最后，子图和多图形窗口也可以直接应用到三维图形函数上。

例程 4.19　plot3 命令示例

```
>> t=0:pi/50:10*pi;
>> plot3(t.*sin(t),t.*cos(t),t);
>> title('Helix'),xlabel('sin(t)'),ylabel('cos(t)'),zlabel('t');     %添加标题，对坐标轴进行标注
>> text(0,0,0,'origin')
>> grid
>> v=axis
v =
-40      40    -40      40       0      40
```

输出结果如图 4-19 所示。

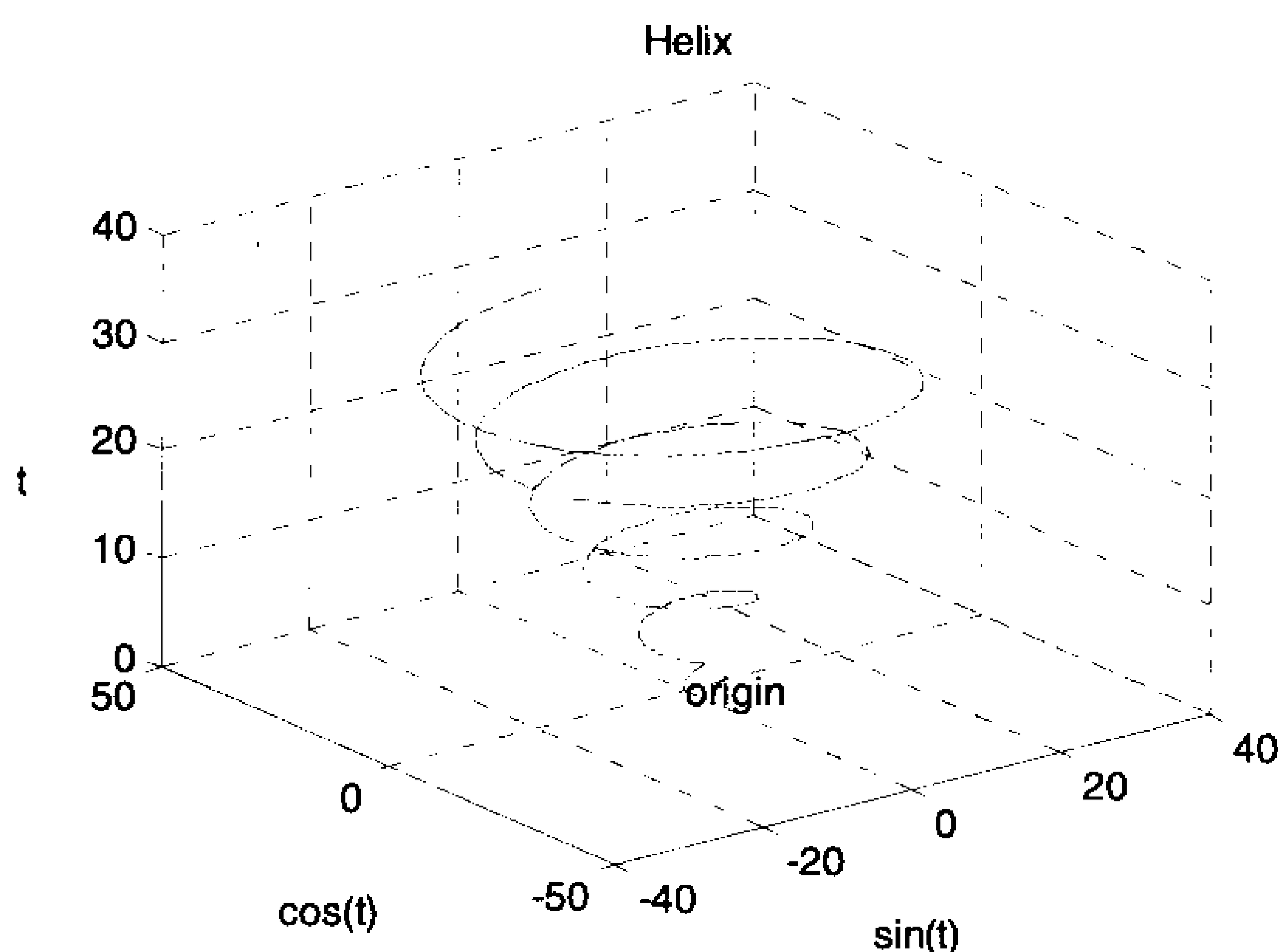


图 4-19　圆锥螺旋曲线

可以明显地看出，二维图形的所有基本特性在三维中仍然存在。axis 命令扩展到三维只是返回 Z 轴界限并在数轴向量中增加两个元素。另外，子图和多图形窗口可以直接应用到三维图形中。

2）三维网格图

所谓网格图，是指把相邻的数据点连接起来形成的网状曲面。利用在 X-Y 平面的矩形网格点上的 Z 轴坐标值，MATLAB 定义了一个网格曲面。三维网格图的形成原理为：在 X-Y 平面上指定一个长方形区域，采用与坐标轴平行的直线将其分格；计算矩形网格点上的函数值，即 Z 轴的值，得到三维空间的数据点；将这些数据点分别用处于 X-Z 或者其平行面内的曲线和处于 Y-Z 或者其平行面内的曲面连接起来，即形成网格图。网格图对于显示大型的数值矩阵很有用处。

建立网格图常用的函数是 mesh，其调用方式如下。

表 4-18　mesh 函数调用格式

调 用 格 式	说　　明
mesh(X,Y,Z,C)	在 X、Y 决定的网格区域上绘制 Z 的网格图。每点的颜色由矩阵 C 决定，若 C 默认，默认颜色矩阵是 C=Z
mesh(Z)	在系统默认颜色和网格区域的情况下绘制数据 Z 的网格图
mesh(Z,C)	在系统默认网格区域的情况下绘制数据 Z 的网格图。颜色由矩阵 C 决定
mesh(…,'ProName','ProVal',…)	绘制三维网格图，并对指定的属性设置属性值

以下为一些网格图的示例及其扩展用法。

例程 4.20　mesh 命令应用示例

```
>> [x,y,z]=peaks;      %返回 MATLAB 三维曲面数据
>> mesh(x,y,z)
```

输出结果如图 4-20 所示。

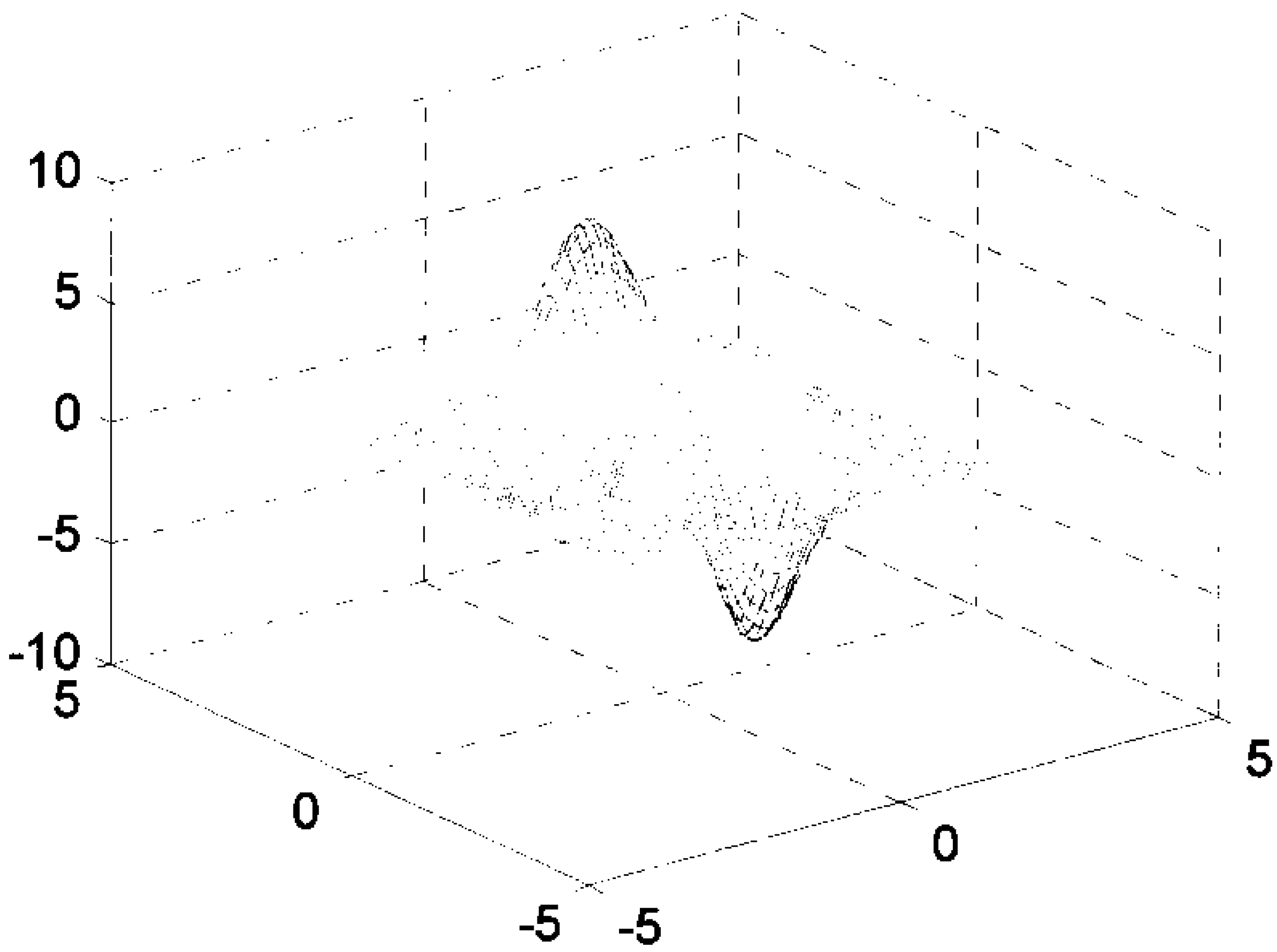


图 4-20　mesh 效果示意图

在彩色显示方式中，可以看到网格图的线条具有不同的颜色，而且颜色随着高度的变化而变化。在任何情况下，如果颜色用于增加图形的有效的第四维空间，这样使用的颜色被称做伪彩色。

在图 4-20 中，我们发现网格线间的区域是不透明的，因此显示的网格只是前面的部分，被遮住的部分没有显示出来。MATLAB 用 hidden 函数控制网格图的这个特性：hidden on 表示隐藏被遮住的部分，hidden off 表示显示遮住的部分。这就是三维图形的透视效应。

还有两个和 mesh 相似的函数：meshc，用于画网格图和基本的等值线图；meshz，用于绘制包含零平面的网格图。

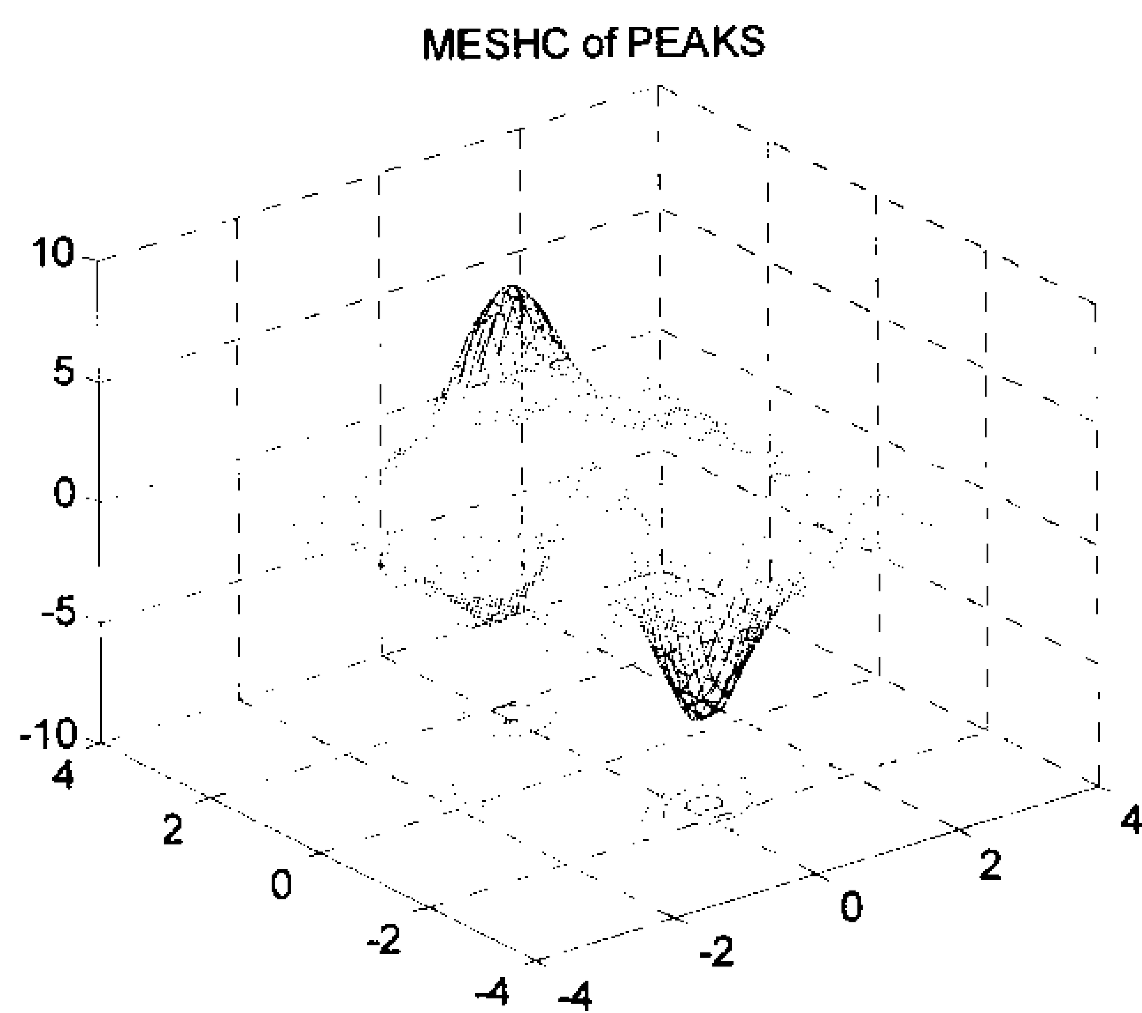
例程 4.21　meshc 和 meshz 函数应用示例

```
>> [x,y,z]=peaks(40);
>> meshc(x,y,z)
>> title('MESHC of PEAKS')
>> hidden off
```

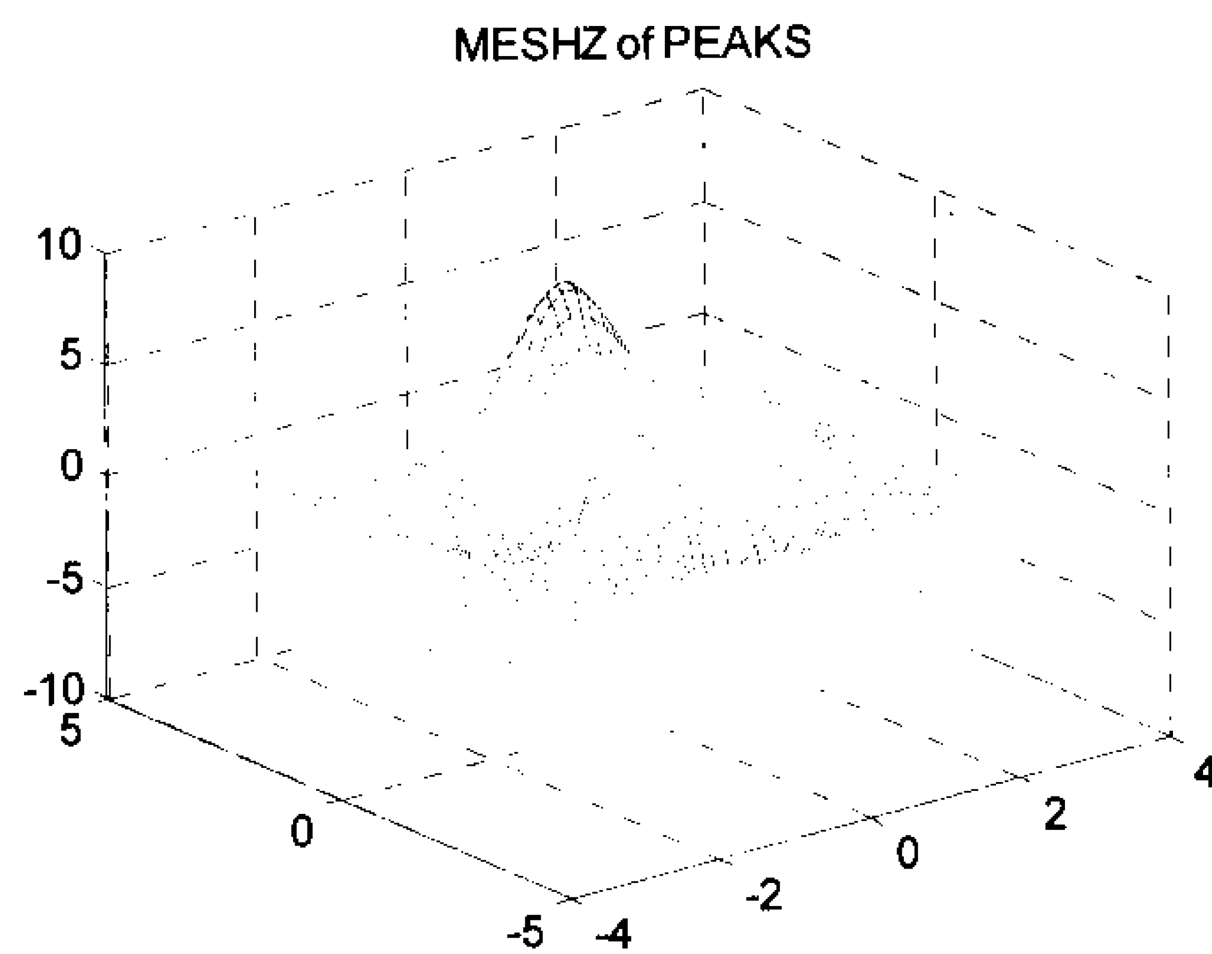
输出结果如图 4-21（a）所示。

```
>> [x,y,z]=peaks(40);
>> meshz(x,y,z)
>> title('MESHZ of PEAKS')
>> hidden on
```

调用函数 hidden on，输出结果如图 4-21（b）所示。



（a）peaks 网格图和基本等值线图



（b）peaks 带零平面的网格图

图　4-21

像二维绘图一样，有时候我们需要在图上用标点来显示某些数值的重要性。关于这个功能，要用到 mesh 与 plot3 两个命令。

例程 4.22　重要数值表示示例

```
>> clear;
>> [x,y]=meshgrid([-3:0.2:3]);        %产生格网矩阵
>> z=peaks(x,y);
>> mesh(x,y,z);
>> hold on;
>> plot3(x,y,z,'x', 'MarkerSize',3);
```

输出结果如图 4-22 所示。

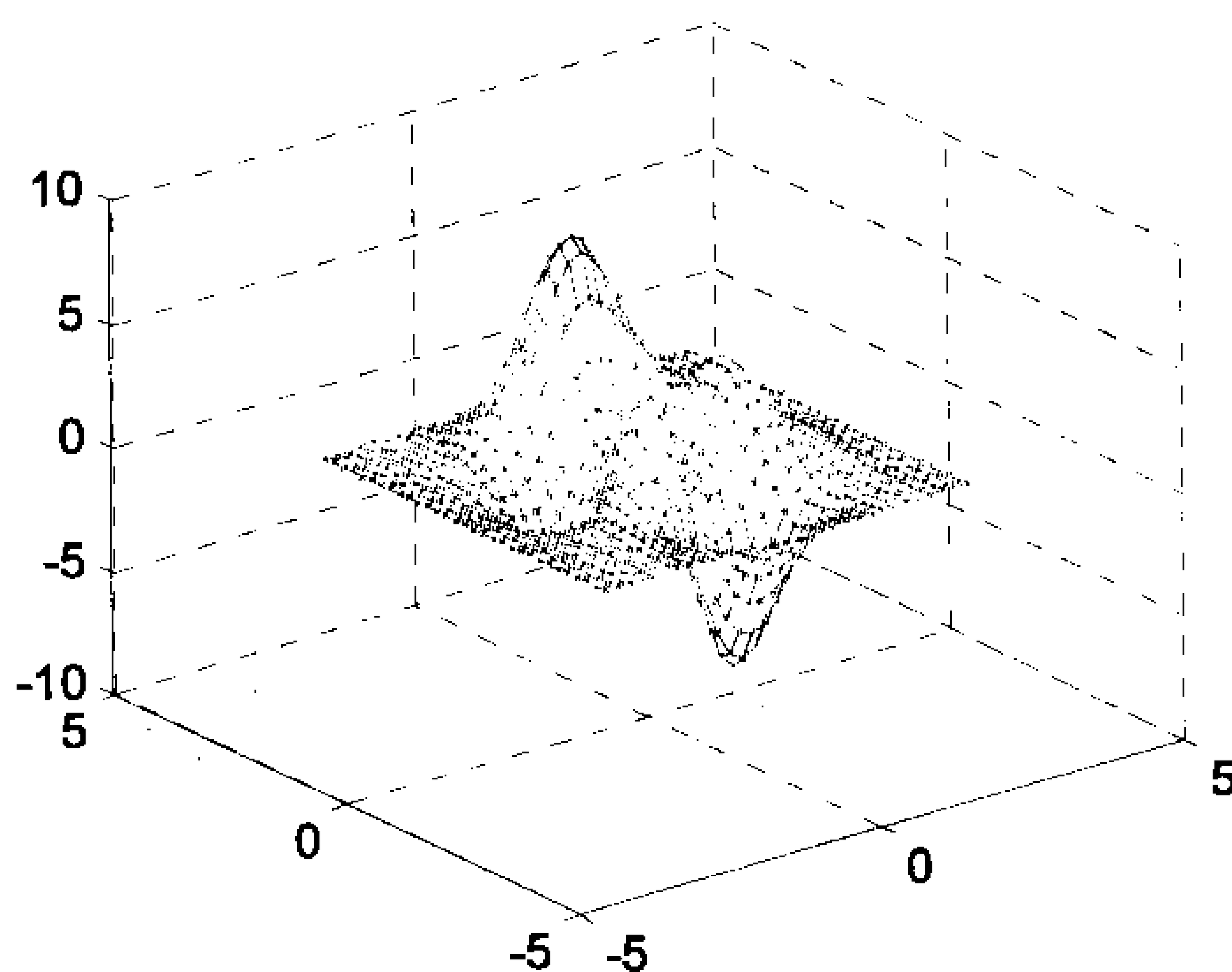


图 4-22　3D 网状图上的标点图

3）三维曲面图

曲面图是把网格图表面的网格围成的小片区域（补片）用不同的颜色填充形成的彩色表面。除了网格线条间的空档用颜色填充外，它和网格图看起来是一样的。用以绘制曲面图的 surf 函数和 mesh 函数的用法完全相同，在此就不再赘述。这两个函数不同的地方就是着色，用 surf 函数建立的图形更具立体感。

例程 4.23　surf 函数应用示例

```
>> [x,y,z]=peaks;
>> surf(x,y,z);
```

输出结果如图 4-23 所示。

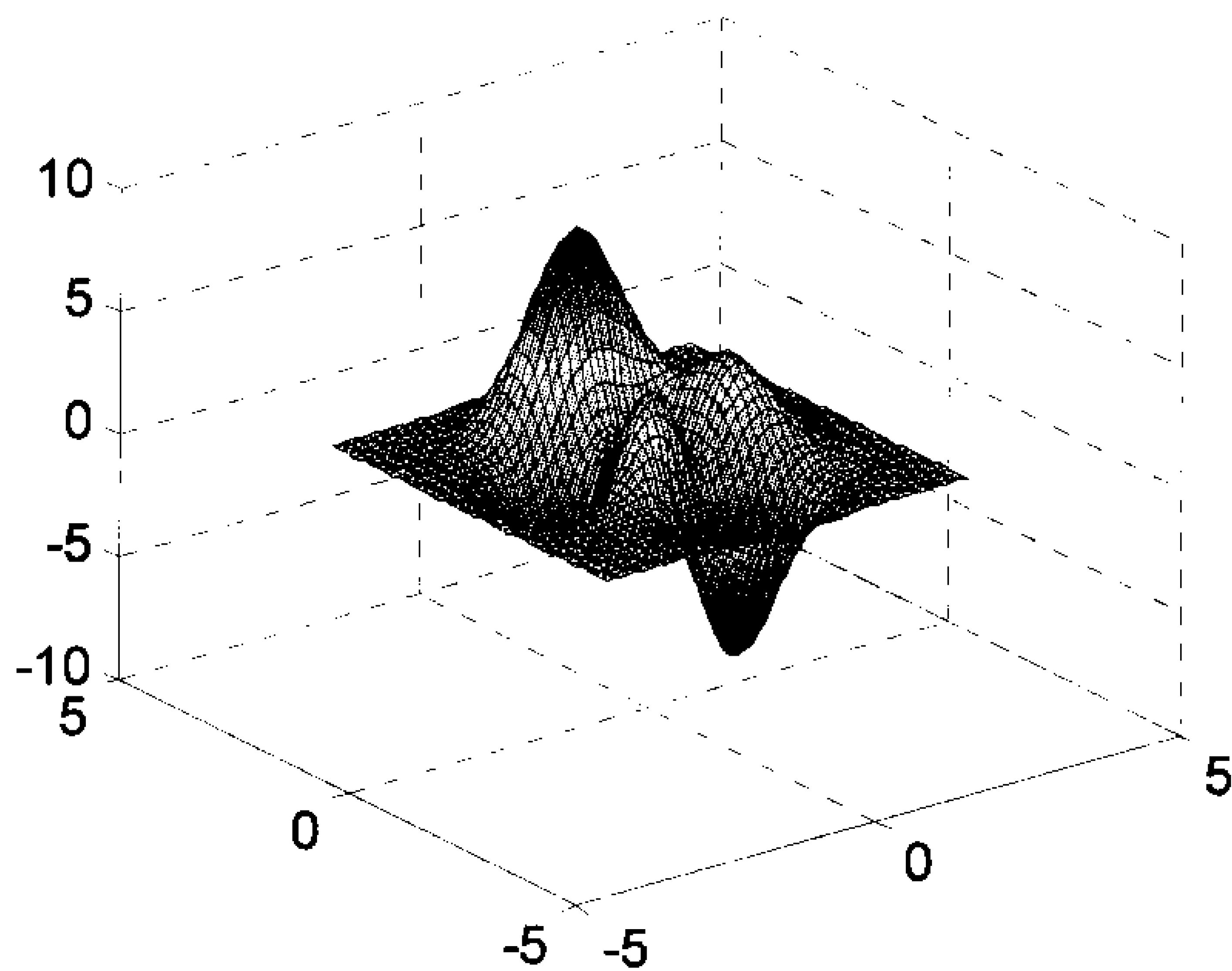


图 4-23　surf 效果示意图

通过结果可以看到，表面图的线条为黑色，补片是彩色，这也是它和网格图的区别所在。

有时想得到表面图的整体效果，其中的线条应该去掉，并可以对颜色做平滑和插值处理，这需要借助函数 shading，它的用法有 3 种，如表 4-19 所示。

表 4-19　shading 函数

调用格式	说　明
shading flat	去掉各片连接处的线条，平滑当前图形的颜色
shading interp	去掉连接线条，在各片之间使用颜色插值，使得片与片之间及片内部的颜色过渡都很平滑
shading faceted	默认值，带有连接线条的曲面，如图 4-23 所示

使用第 2 种内插加色彩，各补片间以插值加颜色，即各补片的颜色根据赋予顶点的值，对其区间进行插值运算。

同 mesh 函数一样，函数 surf 也有两个类似的函数：surfc 用以画出基本等值线的曲线图，surfl 用以画出一个有亮度的曲线图。surfc 函数与 surfl 函数的不同之处在于它是根据光照模型进行着色的，其着色原理是：将环境光、散射光、镜面反射光和漫射光混合在一起作为网格表面的颜色。

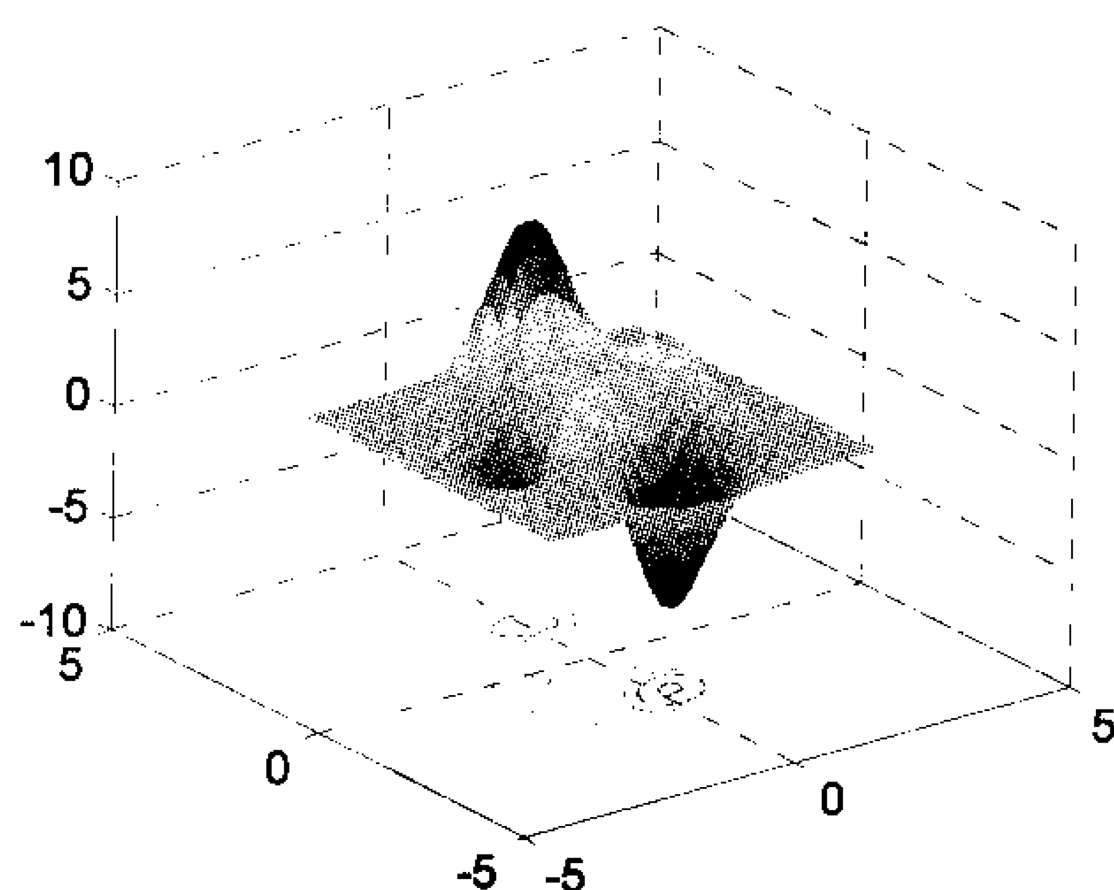
例程 4.24　函数 surfc 和函数 surfl 应用示例

```
>> [x,y,z]=peaks(30);
>> surfc(x,y,z)
>> shading flat
```

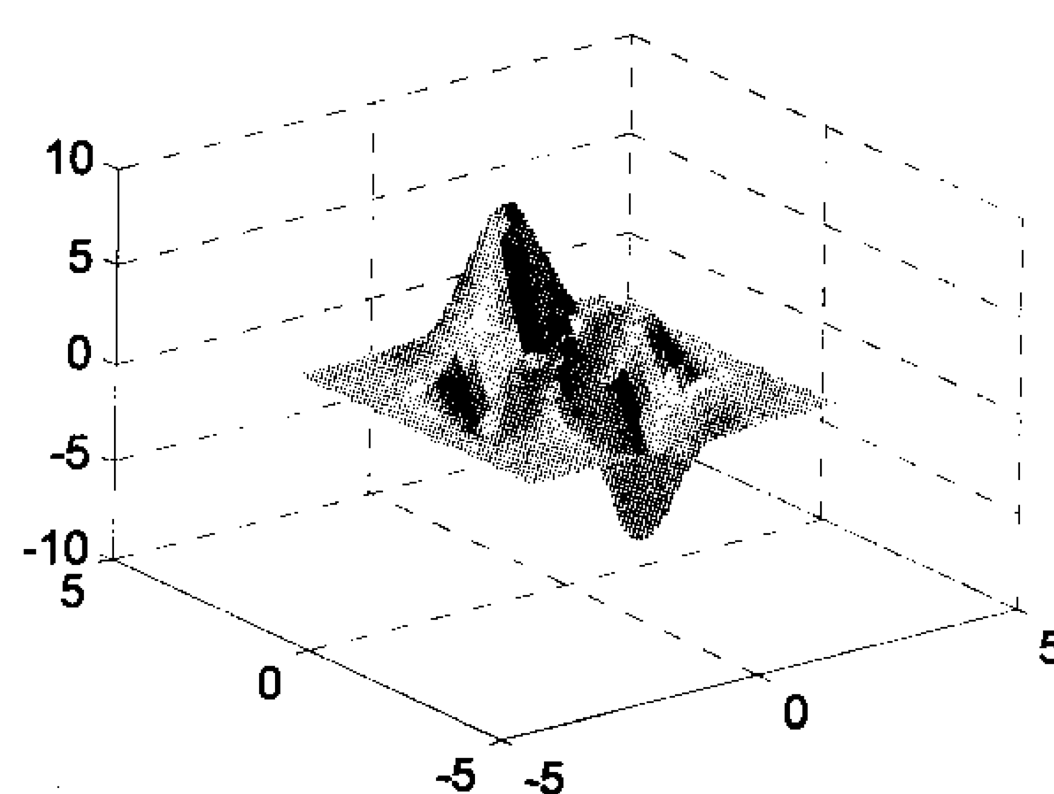
输出结果如图 4-24（a）所示。

```
>> [x,y,z]=peaks(30);
>> surfl(x,y,z)
>> shading interp
```

输出结果如图 4-24（b）所示。



（a）surfc 效果示意图



（b）surfl 效果示意图

图　4-24

MATLAB 还提供了一个建立表面对象的低级函数 surface。该函数用于把表面对象添加到当前坐标系中，以下命令的执行结果和调用 surf 函数是一样的，可以生成彩色表面图。

4.2.2　特殊三维绘图

除了前面讨论的绘图函数以外，MATLAB 还提供了一些专用的三维绘图函数。

1）三维条形图与柱形图

绘制三维条形图使用 bar3 和 bar3h 函数。同二维绘图相似，bar3 用于绘制垂直的条形图，bar3h 用于绘制水平的条形图。绘制柱形图使用 cylinder 函数。它们的调用格式如表 4-20 所示。

表 4-20　bar3、bar3h、cylinder 函数调用格式

调 用 格 式	说　明
bar(z)	绘制 *z* 的三维条形图
bar3(y,z)	在参数向量 *y* 指定的位置绘制三维条形图
bar3(…,width)	在参数向量 *width* 指定的宽度绘制三维条形图
bar3(…,'Style')	在参数向量 *Style* 条件下绘制三维条形图。其中 *Style* 可以为 *'detached'*、*'grouped'* 和 *'stacked'*
bar3(…,LineSpec)	在参数向量 *LineSpec* 指定的线型要素绘制三维条形图
h=bar3(…)	返回条形图的句柄属性值向量
bar3h(…)	绘制水平三维条形图
h=bar3h(…)	返回水平三维条形图的句柄属性值向量
[x,y,z]=cylinder	给出半径为 1 的圆柱形图的 *x*、*y*、*z* 坐标，在每个圆周上取 20 个等间距点
[x,y,z]=cylinder(r)	给出参数 *r* 确定轮廓线的柱形图的 *x*、*y*、*z* 坐标，在每个圆周上去 20 个等间距点

（续表）

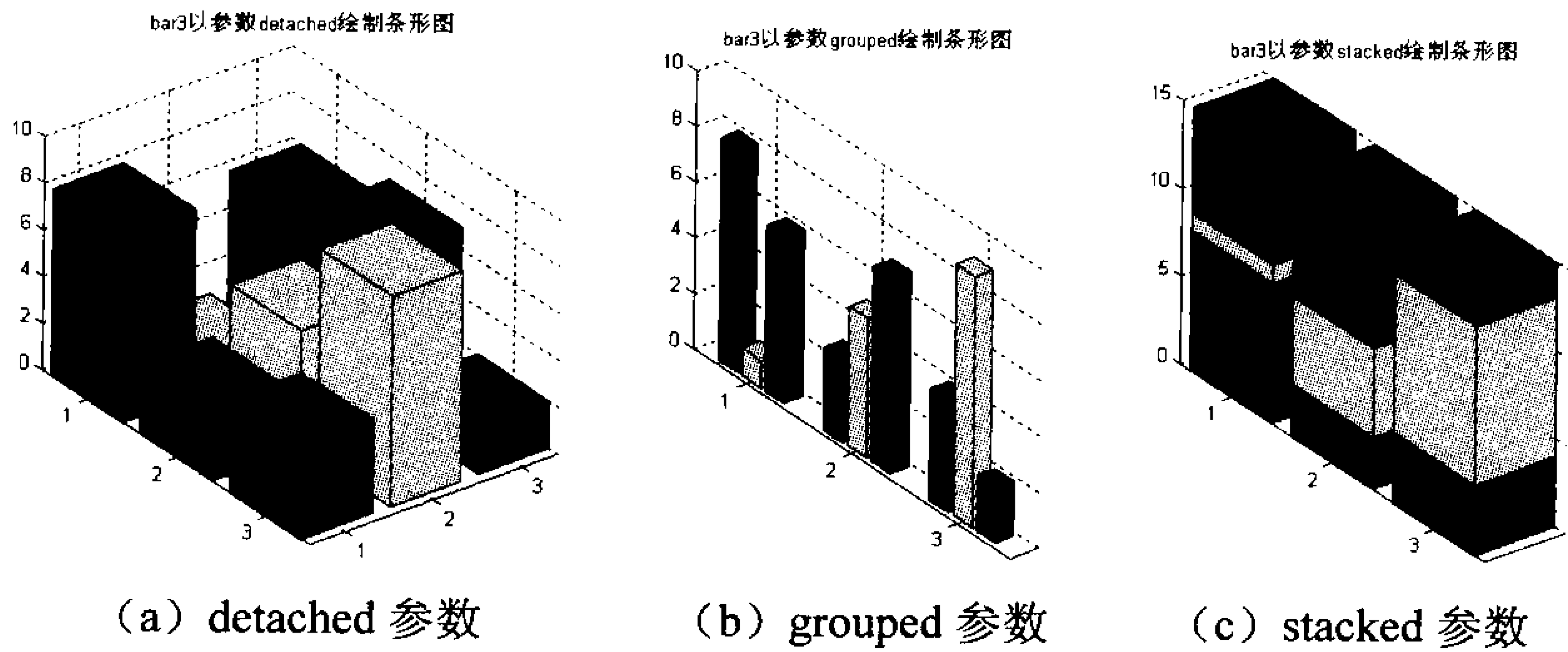
调用格式	说明
[x,y,z]=cylinder(r,n)	给出参数 *r* 确定轮廓线的柱形图的 *x*、*y*、*z* 坐标，在每个圆周上去 *n* 个等间距点
cylinder(…)	不输出参量，用 surf 绘制柱形图

例程 4.25 在各种 *style* 参数的条件下绘制矩阵 *A* 的三维条形图。

例程 4.25　三维条形图绘制示例

```
>> A=magic(3);
>> bar3(A,'detached')
>> title('bar3 以参数 detached 绘制条形图');
>> figure,bar3(A,'grouped')
>> title('bar3 以参数 grouped 绘制条形图');
>> figure,bar3(A,'stacked')
>> title('bar3 以参数 stacked 绘制条形图');
```

输出结果如图 4-25 所示。



（a）detached 参数　（b）grouped 参数　（c）stacked 参数

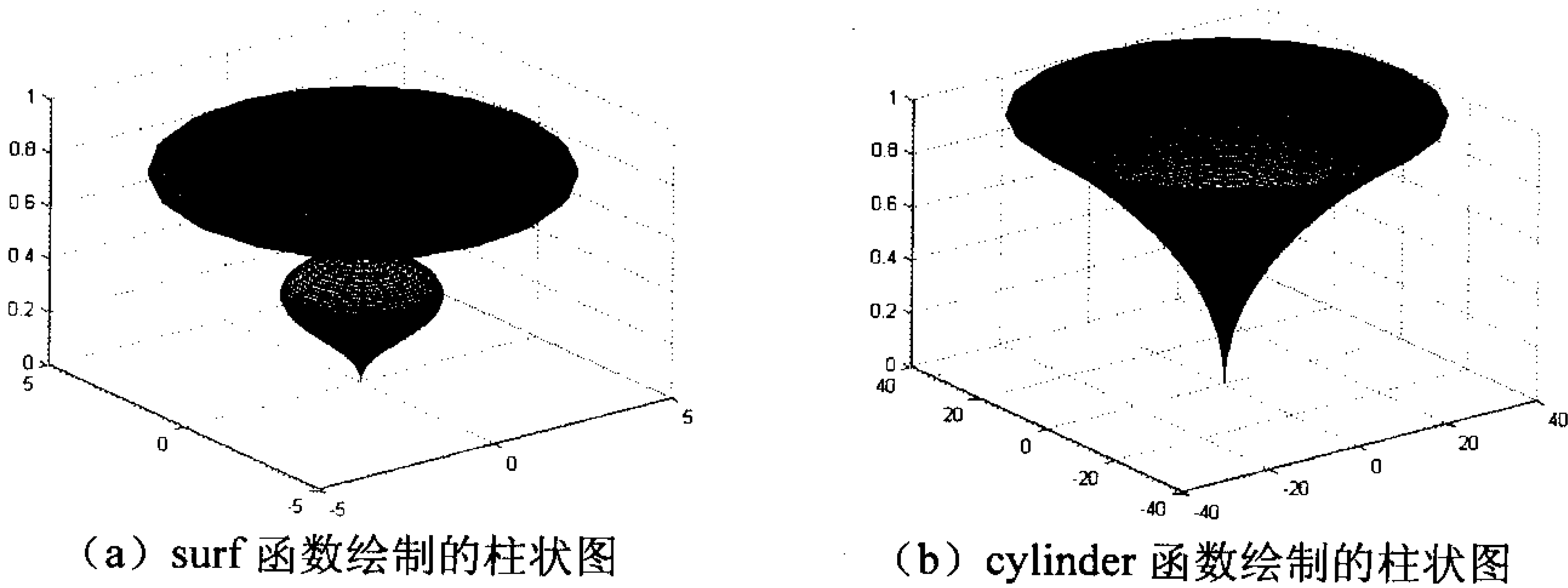
图 4-25　用 bar3 绘制三维条形图

例程 4.26 为三维柱状图绘制示例。

例程 4.26　柱状图绘制示例

```
>> t=[0:pi/50:2*pi];
>> [x,y,z]=cylinder(t.*sin(t));
>> surf(x,y,z)
>> figure,cylinder(t.^2)
```

得到的结果如图 4-26 所示。



（a）surf 函数绘制的柱状图　（b）cylinder 函数绘制的柱状图

图 4-26　柱状图

2）三维火柴杆图与瀑布图

绘制三维离散数据的火柴杆图使用 stem3 函数，该函数用一条线段显示数据离开 *XOY* 面的高度，在线段的顶端用一个小圆圈（默认点型）或其他点型标记高度。

绘制瀑布图使用 waterfall 函数，该函数仅绘制数据 *z* 行数据的直线或曲线，而不绘制列数据的直线，以此显示瀑布效果。stem3 和 waterfall 函数的调用格式如表 4-21 所示。

表 4-21　bar3、bar3h、cylinder 函数调用格式

调用格式	说　明
stem3(Z)	绘制数据序列 *Z* 的火柴杆图
stem3(X,Y,Z)	在 *X*、*Y* 确定的位置上绘制 *Z* 的火柴杆图
stem3(…,'filled')	绘制数据的火柴杆图。参数 *'filled'* 确定是否填充顶端的小圆圈
stem3(…,LineSpec)	用参数 *LineSpec* 确定的线型要素绘制数据的火柴杆图
h=stem3(…)	返回所绘制火柴杆图的句柄属性值向量
waterfall(Z)	绘制数据 *Z* 的瀑布图
waterfall(X,Y,Z)	用所给的 *X*、*Y*、*Z* 数据绘制三维瀑布图
waterfall(…,C)	以参数矩阵 *C* 确定的颜色绘制三维瀑布图
h=waterfall(…)	返回三维瀑布图的句柄属性值向量

例程 4.27 利用 stem3 绘制三维火柴杆图，并修改线条宽度与记号颜色等属性。

例程 4.27　三维火柴杆图绘制示例

```
th=(0:127)/128*2*pi;
x=cos(th);y=sin(th);
f=abs(fft(ones(10,1),128));
h=stem3('v6',x,y,f,'*');%以旧的格式建立 stem3
%由返回的 line 图形对象句柄值来设置线条宽度为 2 与记号变换颜色为红色
set(h,'LineWidth',2,'MarkerEdgeColor','r')
```

输出结果如图 4-27 所示。

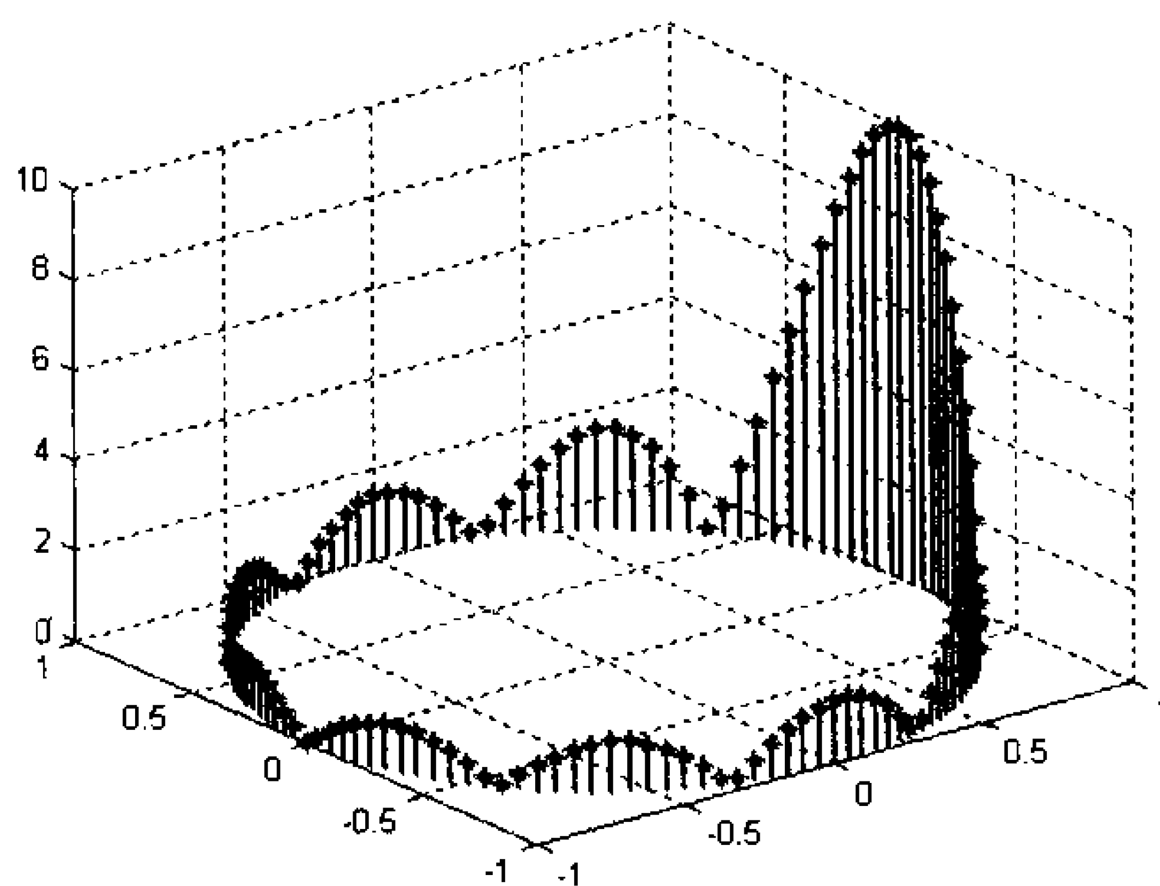


图 4-27　用 stem3 绘制三维火柴杆图

例程 4.28 绘制高斯分布函数的三维瀑布图。

例程 4.28　三维瀑布图绘制示例

```
>> [x,y]=meshgrid(-4:0.05:4);
>> z=peaks(x,y);
```

```
>> waterfall(x,y,z)
```

输出结果如图 4-28 所示。

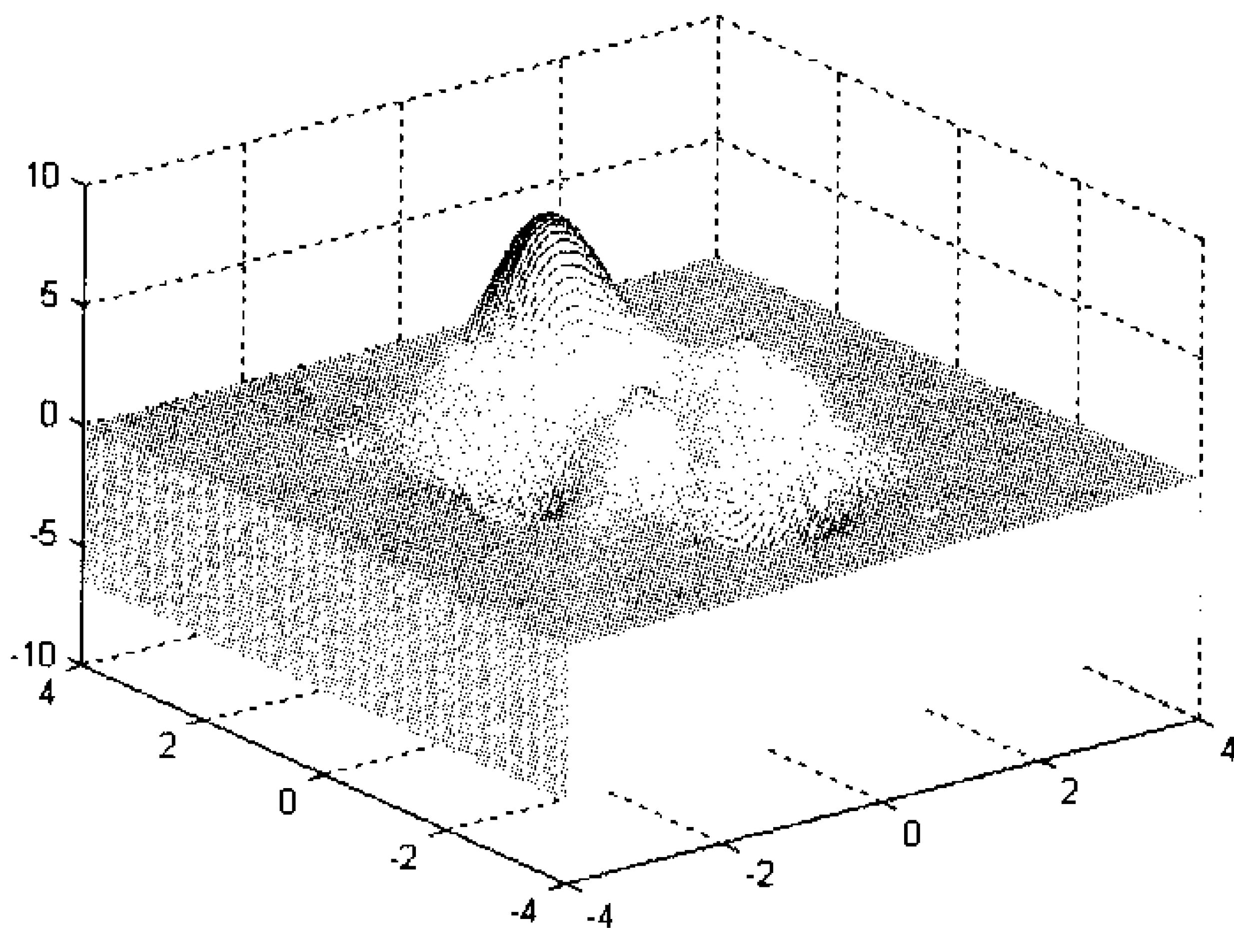


图 4-28　三维瀑布图

3）等高线图和球面图

在 MATLAB 中，函数 contour3 用于绘制三维等高线图，该函数绘制一个定义在矩形栅格上的曲面的等高线图。

绘制球面图使用 sphere 函数。函数 contour3 和 sphere 的调用格式如表 4-22 所示。

表 4-22　contour3 和 sphere 函数调用格式

调 用 格 式	说　明
contour3(Z)	绘制矩阵 Z 的三维等高线图
contour3(Z,n)	绘制具有 n 条等高线的矩阵 Z 的等高线图
contour3(Z,v)	在参数 v 指定的高度上绘制 Z 的等高线图。等高线条数为 length(v)
contour3(…,LineSpec)	以参数 *LineSpec* 指定的线型要素绘制三维等高线图
contour3(X,Y,Z)、contour3(X,Y,Z,n) contour3(X,Y,Z,v)	用 X、Y 确定的 X、Y 轴的范围绘制 Z 的等高线图
[C,h]=contour3(…)	返回等高线矩阵 C 及其句柄属性值向量
sphere	建立一个由 20×20 个面组成的球面
sphere(n)	建立一个由 n×n 个面组成的球面
[x,y,z]=sphere(…)	不绘制球面，返回球面的坐标矩阵

以下为绘制等高线图和球面图的示例。

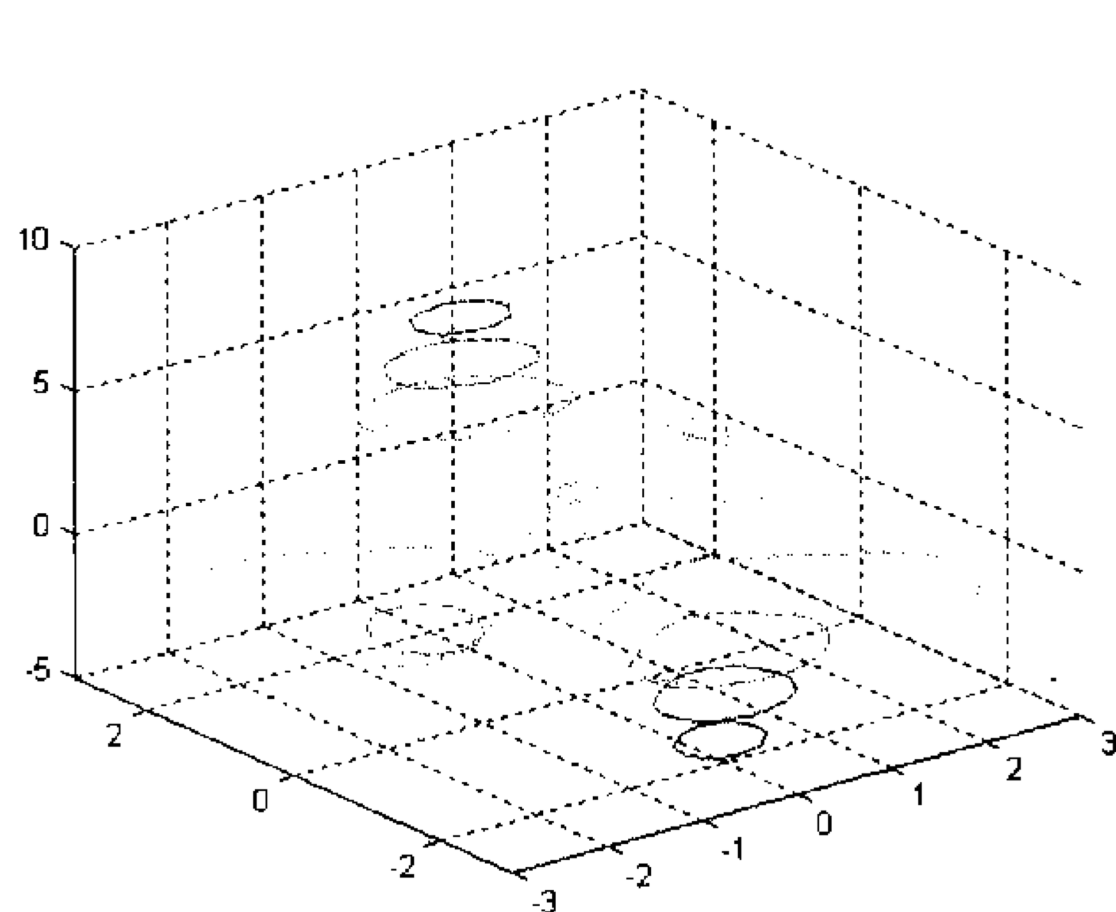
例程 4.29　等高线绘制示例

```
>> [x,y,z]=peaks(30);
>> [C,H]=contour3(x,y,z,16);
```

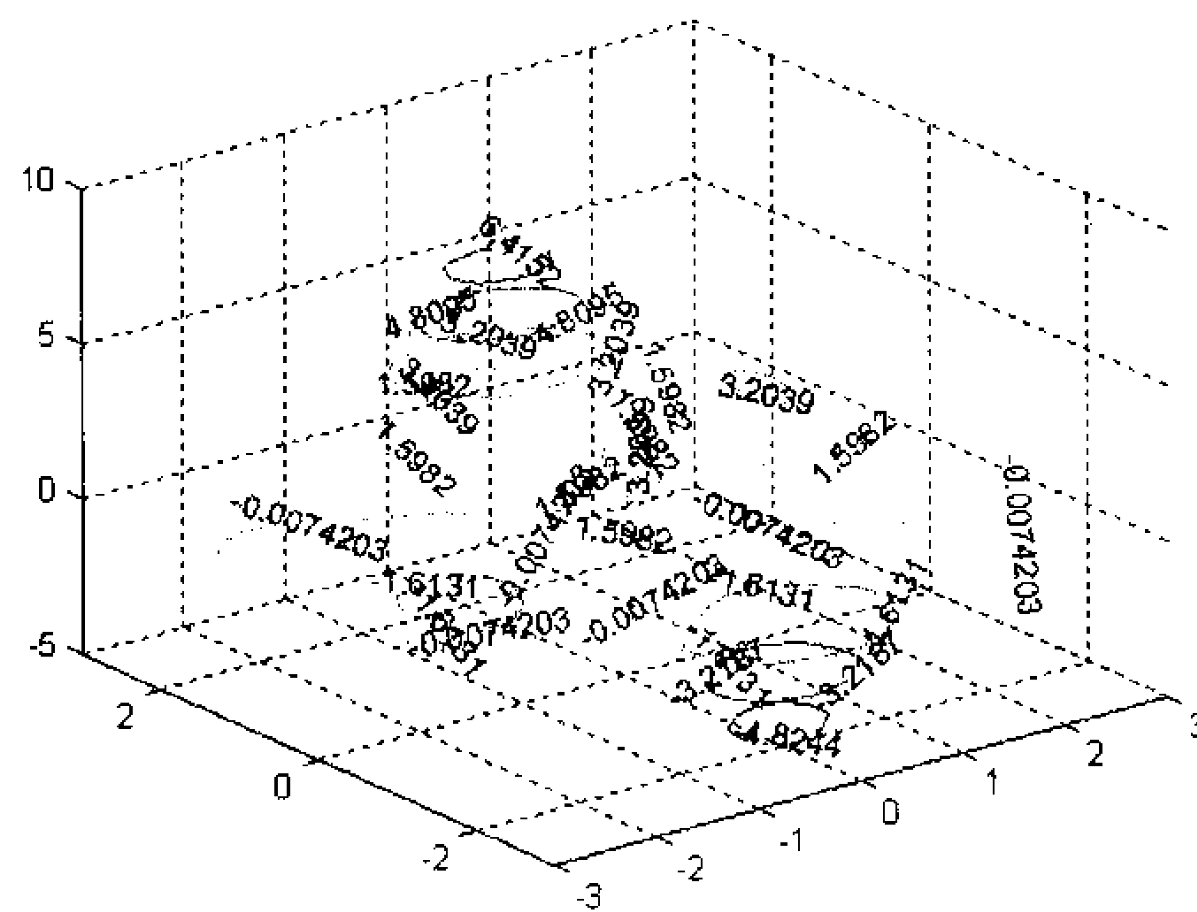
输出结果如图 4-29（a）所示。绘制好的等高线可以用 clabel 函数标注高度值。在例程 4.27 中添加如下命令：

```
>> clabel(C,H)
```

得到结果如图 4-29（b）所示。



（a）未标注三维等值图



（b）标注后三维等值图

图 4-29

例程 4.30 实现的是：随即输入欲绘制的球体数后，以随机数产生的方式来绘制球体。

例程 4.30　利用 sphere 进行绘图

```
N=input('请输入要绘制的圆数目：');
[X,Y,Z]=sphere;
for k=1:N
        R=rand/8;    %使用随机产生的半径
        Origin=rand(1,3);    %使用随机产生的初始位置，也就是圆点中心位置
        surf(X*R+Origin(1),Y*R+Origin(2),Z*R+Origin(3));
        hold on;
end
axis equal
set(gca,'projection','perspective')    %产生有点远视观察的立体技巧
```

输出结果如图 4-30 所示。

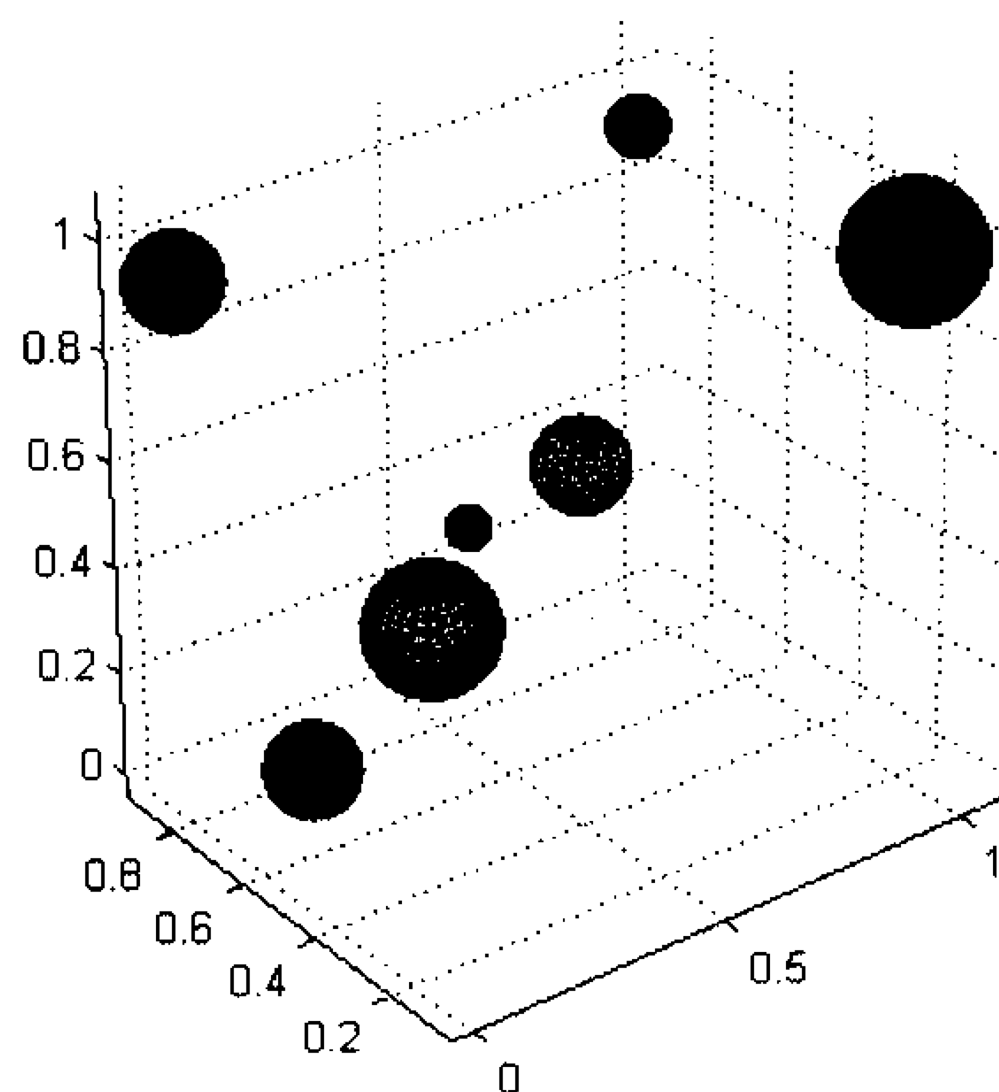


图 4-30　sphere 绘图结果

4）三维向量图

在 MATLAB 中使用 quiver3 函数可以绘制出平面上的向量图或速度图，也就是在等值线图上画出方向或速度箭头，因此非常适合表示分布于平面的向量场，如平面的流速分布图，调用格式如表 4-23 所示。

表 4-23　quiver3 函数调用格式

调用格式	说　明
quiver3(X,Y,Z,U,V,W)	在位置(x,y,z)处绘制元素(u,v,w)的向量图。其中 X、Y、Z、U、V、W 具有相同的大小
quiver3(Z,U,V,W)	在矩阵 Z 确定的等间距表面上绘制向量图，向量的显示比例由它们之间的距离决定
quiver3(…,S)	绘制矢量图，向量的显示比例由它们之间的距离乘以参数 S 决定的系数
quiver3(…,LineSpec,'filled')	由参数 *LineSpec* 决定线型要素，并由参数*'filled'*决定是否填充
quiver3(*…,LineSpec)*	绘制由参数 *LineSpec* 决定线型要素的向量图

例程 4.31 为使用 quiver3 进行绘图示例。

例程 4.31　三维向量图绘制示例

```
>> [X,Y,Z]=peaks(20);
>> [Nx,Ny,Nz]=surfnorm(X,Y,Z);    %求空间表面的法线
>> surf(X,Y,Z)
>> hold on
>> quiver3(X,Y,Z,Nx,Ny,Nz)
>> axis tight
>> hold off
```

输出结果如图 4-31 所示。

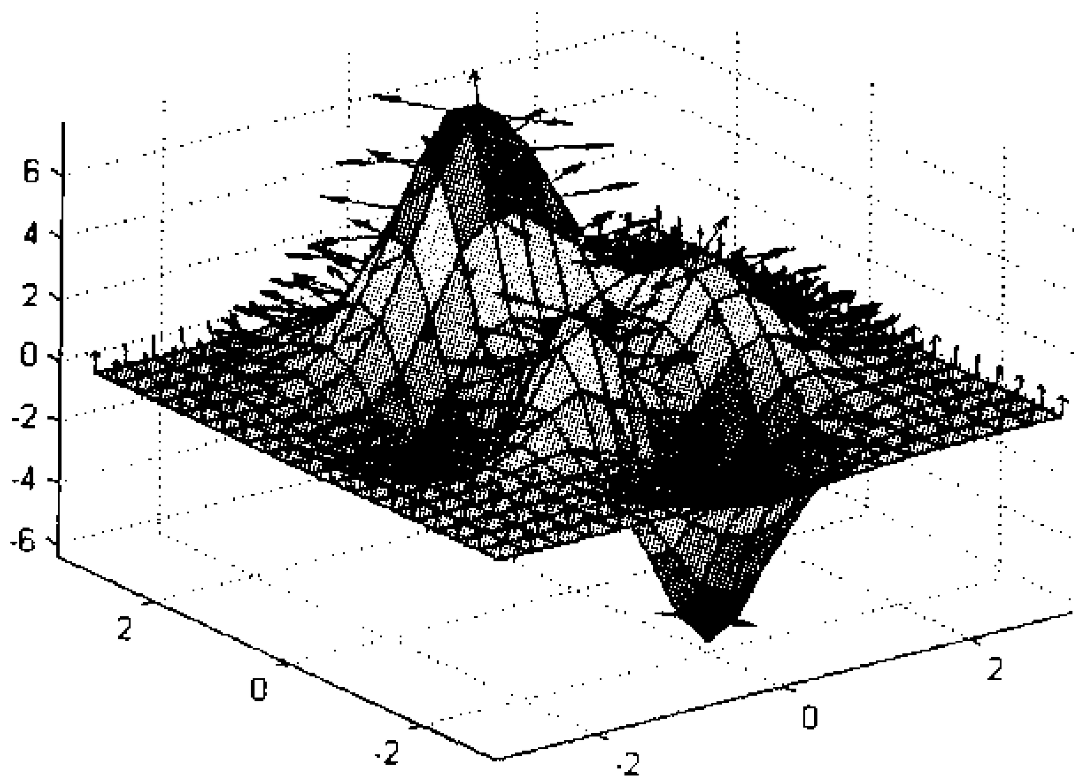


图 4-31　三维箭头图

5）三角网目图

三角网目图在实际工程中应用得比较多，如绘制高程面等。在 MATLAB 中，可以用 trimesh 绘制三角网目图，调用格式如表 4-24 所示。

表 4-24　trimesh 函数调用格式

调用格式	说　明
trimesh(TRI,X,Y,Z,C)	由 *TRI* 所定义的 $M\times3$ 矩阵的三角数据来绘制三角网目图
trimesh(TRI,X,Y,Z)	绘制由参数 C 确定颜色的三角网目图
trimesh(TRI,X,Y)	在二维绘图中显示三角图
h=trimesh(…)	返回显示的三角网目图的句柄值
trimesh(…,'ProName','ProVal',…)	绘制三角网目图，但对*'ProName'*指定属性设置属性值

试绘制下列函数的三角网目图。

$$z=\frac{\mathrm{e}^{-\sin R}}{R}，\quad R=\sqrt{x^2+y^2}$$

例程 4.32　利用 trimesh 进行绘图

```
[x,y]=meshgrid(1:15,1:15);
R=sqrt(x.^2+y.^2);
z=exp(-sin(R))./R;
tri=delaunay(x,y);        %返回三角剖分数据
trimesh(tri,x,y,z);
```

输出结果如图 4-32 所示。

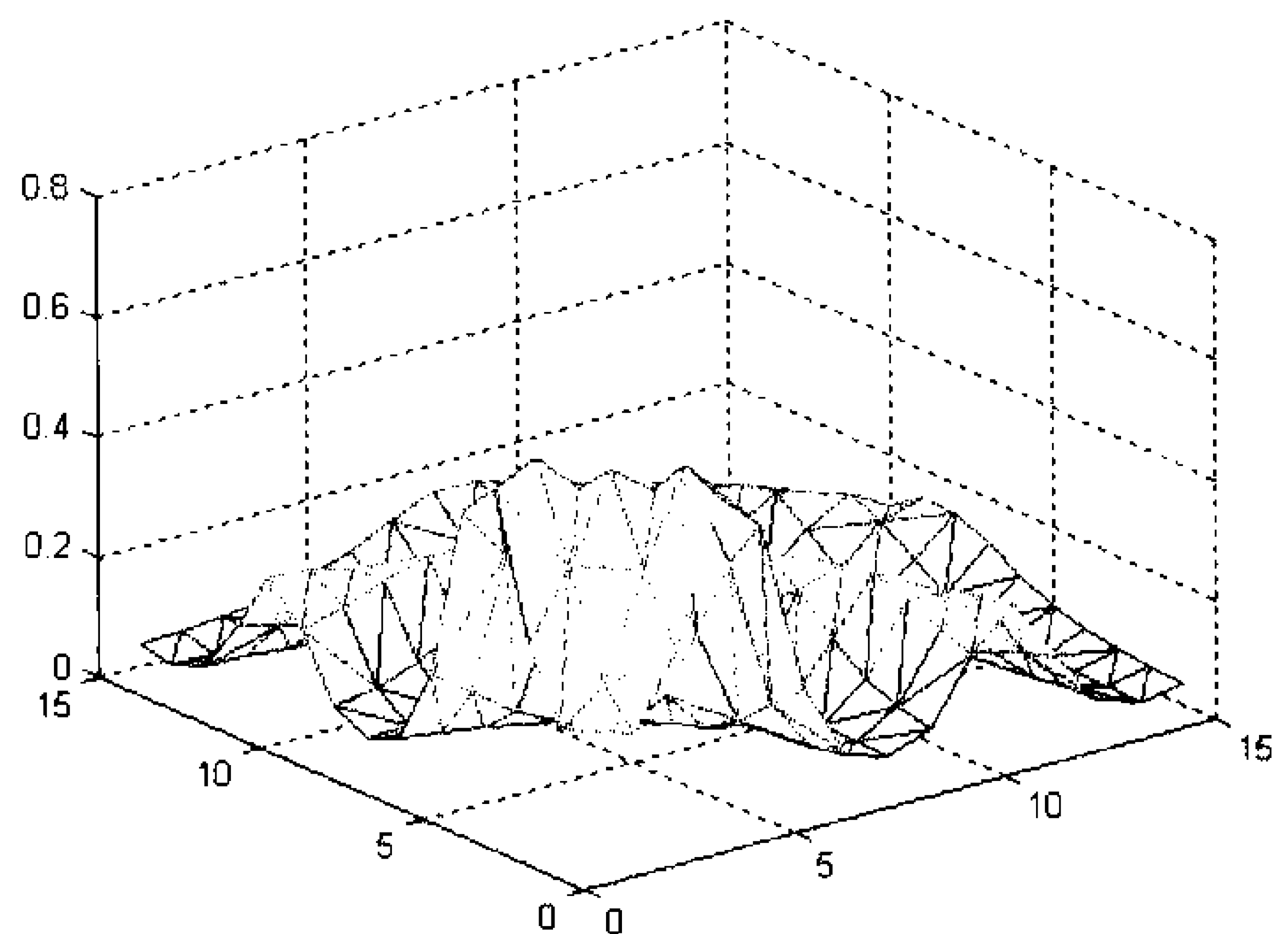


图 4-32　trimesh 绘图结果

MATLAB R2007 中的三维绘图函数还有三维彗星图绘制函数 comet3、三维切片图绘制函数 slice 等，它们的调用格式与前面介绍的其他三维绘图函数都大同小异，限于篇幅，在此就不再一一说明。

4.2.3　三维绘图功能进阶

这里介绍三维图形绘制上的一些进阶技巧，包含设置观看视角、曲面裁切、光源设置及材质处理等。

1．视角改变

所谓视角，简单地讲就是观察（显示）图形的方向，调整视角可以使得一幅图显示出来自不同方向的观察结果。在 MATLAB 中，函数 view 改变所有类型的二维和三维图形的图形视角。它的调用格式如表 4-25 所示。

表 4-25　view 函数调用格式

调 用 格 式	说　明
view(az,el)	设置观察图形的视角
view([az,el])	设置观察图形的视角
view([vx,vy,vz])	通过直角坐标设置视点
view(n)	设置默认的 n 维视角，n 为 2 或 3
[az,el]=view	返回当前的视角
view(T)	用一个 4×4 的转置矩阵 T 来设置视角
T=view	返回当前的 4×4 转置矩阵

其中 *az* 表示方位角（Azimuth），单位是度；*el* 表示视角（Elevation），单位是度；*vx*，*vy*，*vz* 是视角的直角坐标。

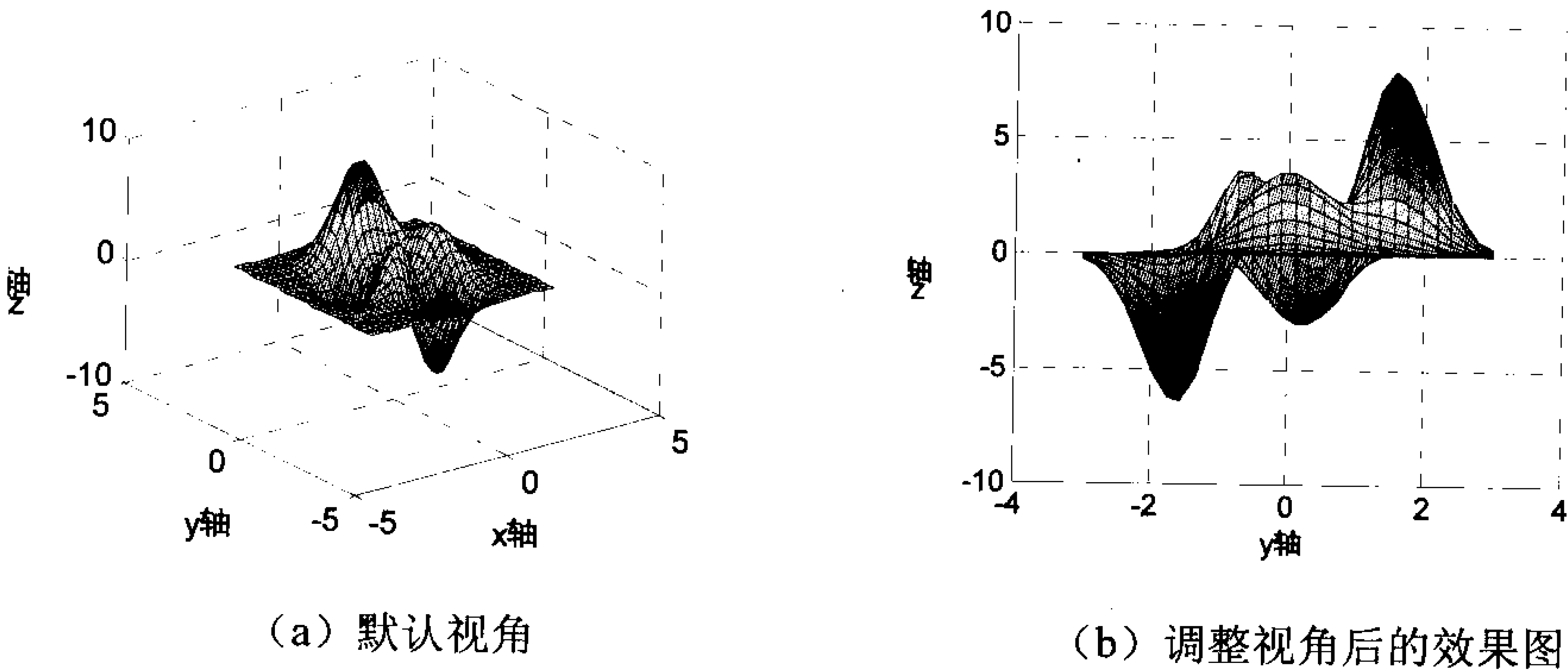
例程 4.33　view 函数使用示例

```
>> [x,y,z]=peaks(30);
>> surf(x,y,z)
>> xlabel('x 轴')
>> ylabel('y 轴')
>> zlabel('z 轴')
```

输出结果如图 4-33（a）所示。

```
>> view(90,0)
```

输出结果如图 4-33（b）所示。



（a）默认视角　　（b）调整视角后的效果图

图 4-33

图 4-33 的视角默认值为：$az=-37.5°$，$el=30°$；通过 view 命令，把方向角在 x-y 平面内从 y 轴负方向逆时针旋转 90°，转到了 x 轴的正方向，而仰角为 0°，即视线从 x 轴的正方向水平看过去的效果。

虽然用函数 view 对三维视角进行控制十分方便，但其功能却十分有限。为了能够对三维场景进行全面控制，就需要用到摄像头功能。当摄影师用一个摄像机拍摄电影时，必须要了解摄像机的全部功能，那么，当用户利用计算机设置三维图形或控制台游戏环境时，也必须了解摄像机的所有功能。在上述环境中，用户通常需要对两个三维坐标系统进行管理——一个是摄像机所在的坐标系，另一个是摄像机所指的坐标系，也就是摄像机目标坐标系。MATLAB 提供了一些摄像机函数用语管理和处理这两个坐标系统之间的关系，并且提供对摄像机镜头的控制，如表 4-26 所示。其函数调用方式可以参见联机帮助。

表 4-26　摄像机函数

函 数 名	功　能	函 数 名	功　能
campos	摄像头位置	camtarget	摄像头目标
camproj	摄像头投影	camva	摄像头视角
camzoom	放大摄像头	camroll	滚动摄像头
campan	固定窗口位置旋转对象	camlookat	查找特定的对象
camup	设置窗口相对于显示对象的位置向量	camlight	生成摄像头光照对象并将其放置在合适的位置

2．曲面裁剪

因为曲面图不能做成透明的，但在一些情况下可以很方便地移走一部分表面图以便看到表面图以下部分。在 MATLAB 中，这是通过所期望的洞孔所在位置，将数据置为特定的 NaN 来实现。由于 NaN 没有任何值，所有的 MATLAB 作图函数都会忽略 NaN 的数据点，在该点出现的地方留下一个洞孔。

例程 4.34 利用 NaN 进行曲面裁剪示例

```
>> [X,Y,Z]=peaks(30);
>> x=X(1,:);y=Y(:,1);
>> i=find(y>0.8 & y<1.2);j=find(x>-0.6 & x<0.5);    %查找符合条件的 i 值
>> Z(i,j)=nan*Z(i,j);
>> surf(X,Y,Z)
>> title('surf   of peaks with a hole')
>> grid on
```

输出结果如图 4-34 所示。

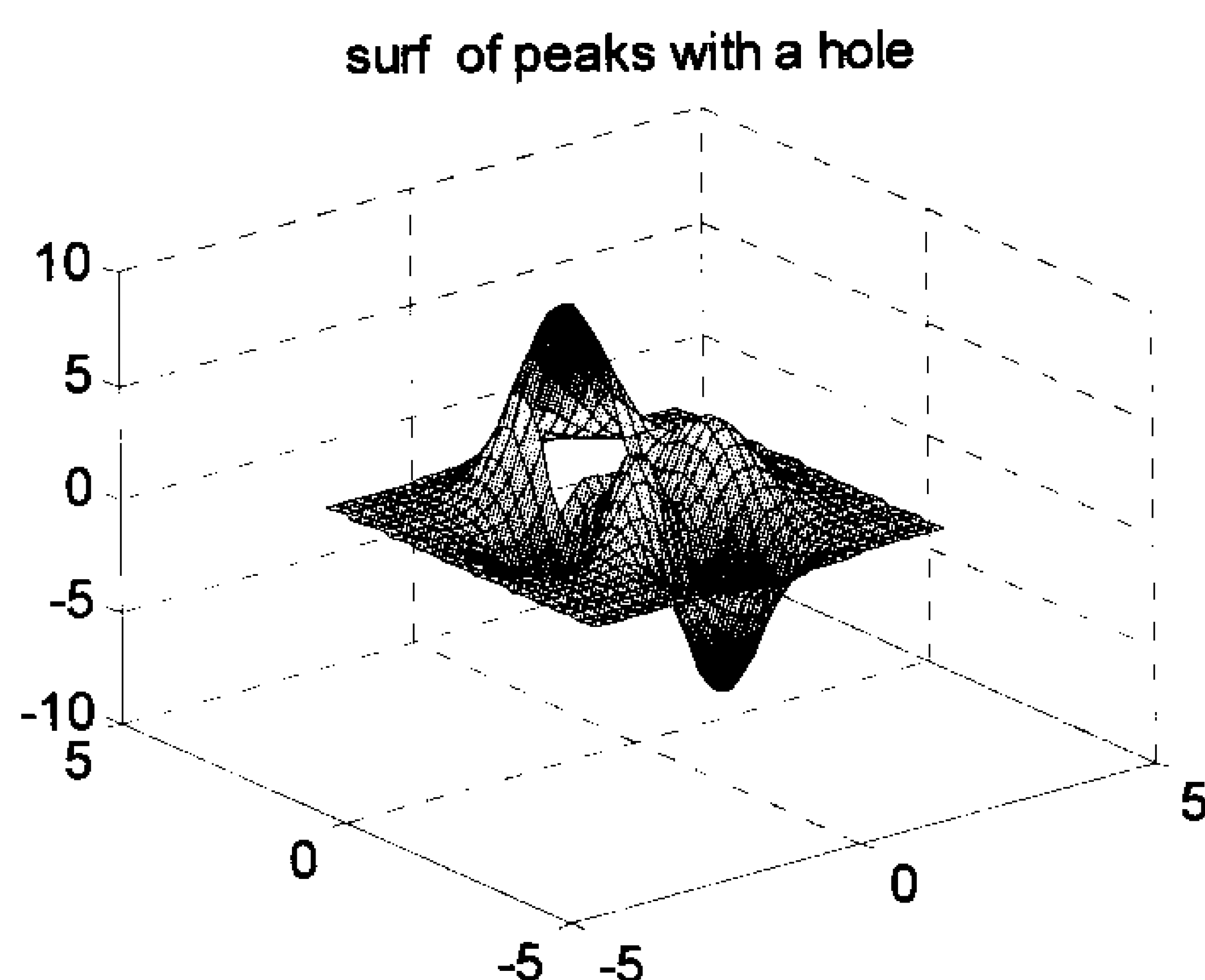


图 4-34 函数 peaks 的带洞孔曲面图

3．光源设置

为图形加上光源能够得到更加逼真的效果。MATLAB 提供了放置光源和调整光照目标特性的一些函数，合理利用这些函数能得到非常真实的视觉效果。光源设置函数如表 4-27 所示。

表 4-27 光源设置函数

函数名称	函 数 功 能	函数名称	函 数 功 能
camlight	根据照相机的位置建立或移动光源	lighting	选择光照方法
lightangle	在球形坐标系中建立或者放置光源	material	设置被照目标的反射特性
light	建立光源对象		

在建立光照效果之前要首先建立光源，可以用 light、camlight 或者 linghtangle 函数来完成。light 函数的调用方式如表 4-28 所示。

表 4-28　light 函数调用格式

调 用 格 式	说　明
light	使用默认值建立光源
h=light(...)	返回建立光源对象的句柄
light(...'ProName', 'ProVal',...)	在建立光源时设置参数

在设置光源前，图形采用的是强度各处相等的漫射光。一旦设置被执行，虽然光源本身并不出现，但图形上的“面”、“轴”、“块”等对象的所有与光有关的属性（如背景光等）都被激活。该命令不包含任何输入参数，则采用默认设置：白光、无穷远、穿过 [1,0,1] 射向坐标原点。

通过函数 lighting options 设置照明模式，该指令只有在 light 命令执行后才能起作用。options 有以下 4 种取值，具体含义如表 4-29 所示。

表 4-29　函数 lighting options 参数含义

参　数	描　述
flat	入射光均匀洒落在图形对象的每个面上，主要与 faceted 配合使用，它是默认形式
gouraund	先对顶点颜色插补，再对顶点勾画的面色进行插补，用于曲面表现
phong	对顶点处的法线插值，再计算各像素的反光，效果好，但费时
none	关闭所有光源

4．材质处理

控制光效果的材质命令为 material，其调用格式如下：

```
material options
```

为方便用户使用，MATLAB 提供了以下 4 种预定义的表面反射模式，即 options 的取值如下。

表 4-30　MATLAB 预定义表面反射模式含义

参　数	描　述
shiny	使对象比较明亮。镜反射份额较大，反射光颜色仅取决于光源颜色
dull	使对象比较暗淡。漫反射份额较大，没有镜面亮点，反射光颜色仅取决于光源颜色
metal	使对象带金属光泽。镜反射额很大，背景光和漫反射额很小。反射光源和图形表面二者的颜色。该模式为默认设置
default	返回默认设置模式

下面通过示例介绍它的应用。

例程 4.35　灯光、照明、材质命令所表现的图形的示例

```
>> clf;                                    %<1>
>> [x,y,z]=sphere(40);                     %<2>
>> colormap(jet)                           %<3>
>> subplot(1,2,1);                         %<4>
>> surf(x,y,z)                             %<5>
>> shading interp                          %<6>
```

```
>> light('position',[0,-10,1.5],'style','infinite')                    %<7>
>> lighting phong                                                       %<8>
>> material shiny                                                       %<9>
>> subplot(1,2,2);                                                      %<10>
>> surf(x,y,z,-z)                                                       %<11>
>> shading flat                                                         %<12>
>> light;lighting flat                                                  %<13>
>> light('position',[-1,-1,-2],'color','y')                             %<14>
>> light('pisition',[-1,0.5,1],'style','local','color','w')             %<15>
```

输出结果如图 4-35 所示。

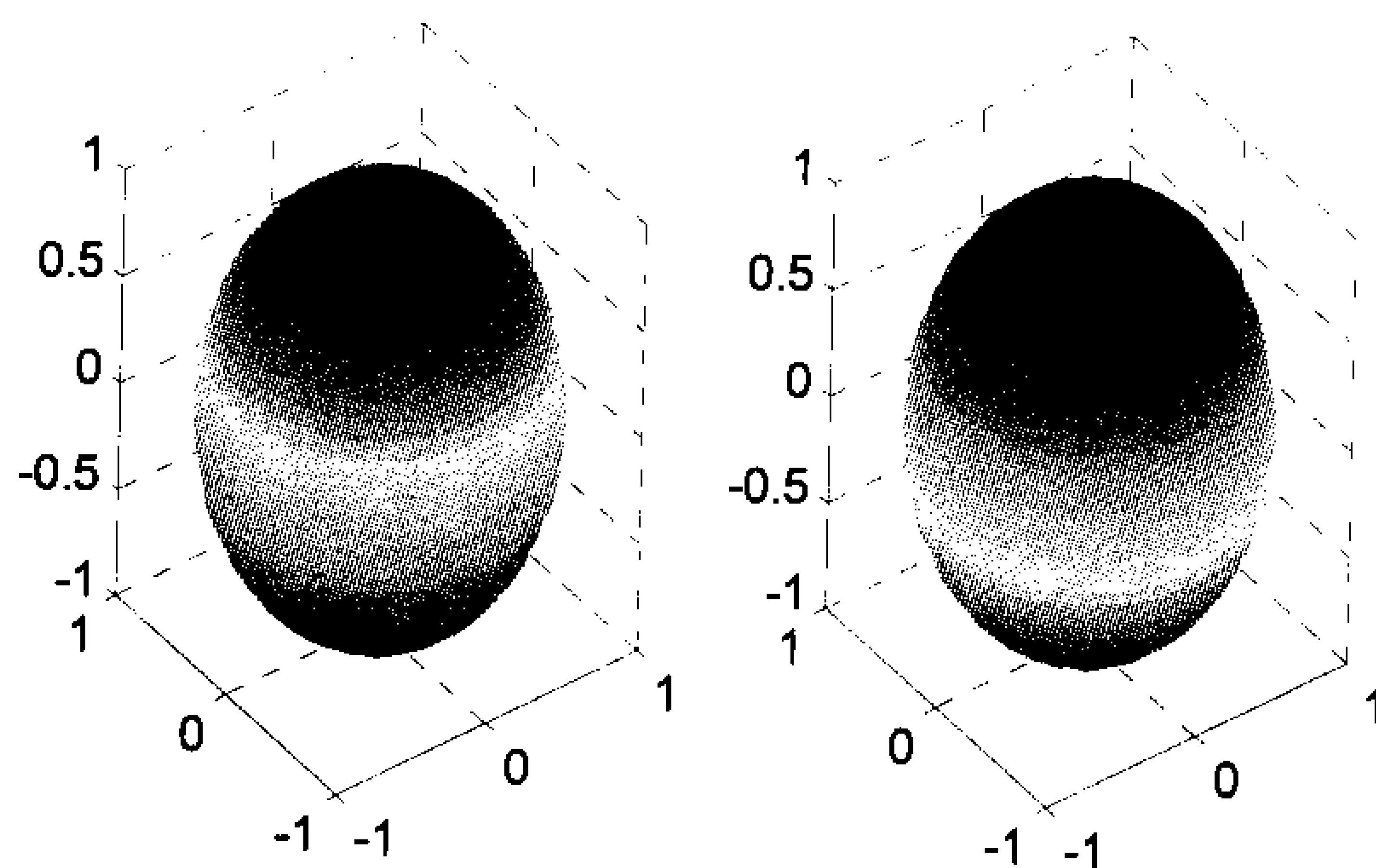


图 4-35　灯光、照明、材质命令所表现的图形

色图是图形窗口的属性。每个图形窗口只有一个色图，见本例指令<3>；每个子图都能定义自己的浓淡处理模式、照明模式和材质，但它们都只能定义一次，如本例左子图相关指令为<4>～<9>，而右子图相关定义指令为<10>～<13>；每个子图上可以设置多个光源，如本例左子图只使用了 1 个默认设置光源，而右子图使用了包括默认设置光源在内的 3 个形式、方向和颜色不同的光源。

4.2.4　透明度作图

在 MATLAB 中使用 hidden 函数控制移除三维图形中显示的隐藏线。隐藏线条的移除其实是显示从视点上看因为被其他物体遮住而模糊不清的线。其调用格式如表 4-31 所示。

表 4-31　hidden 函数使用方法

调用格式	说　明
hidden on	对当前图形打开隐藏线移除的状态，因此三维图后方的线会被前面的线遮住，简单来说就是会透视被叠压的图形
hidden off	对当前图形关闭隐藏线移除的状态，因此三维图后方的线将不会被前面的线遮住，也就是说该三维图会变成一个透明的图
hidden	切换 hidden 为 on 或 off 的状态

例程 4.36 绘制了一个三维图形，一个球体包住网目图，并且将球体设置为透明。

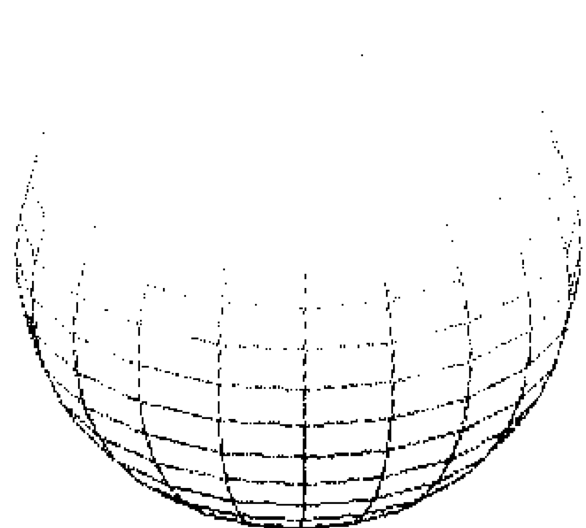
例程 4.36 函数 surfc 和函数 surfl 应用示例

```
[x,y,z]=sphere(20);
x=8.2*x;y=8.2*y;z=8.2*z;        %设置球体的 x、y 与 z
peaks;shading interp;           %使用 interp 渲染方式
colormap(hot);                  %使用 hot 颜色映射值
hold on,mesh(x,y,z),hold off    %以 mesh 来绘制球体数据
axis equal                      %产生当长的坐标轴以便于球体的显示
axis off                        %将坐标轴隐藏
```

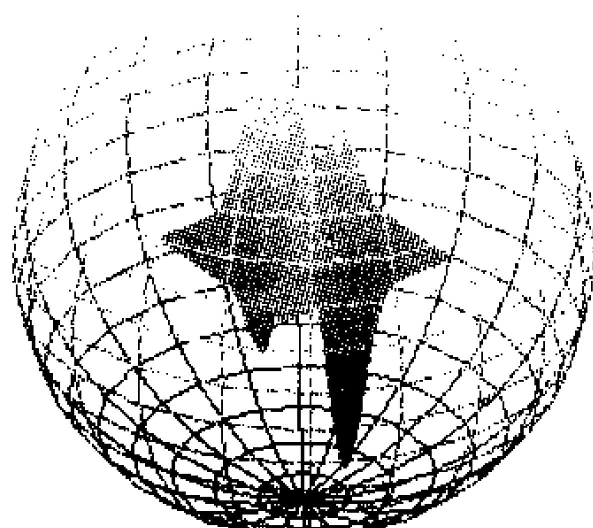
输出结果如图 4-36（a）所示。现在在例程 4.36 中加入以下语句：

```
hidden off    %将球体设置为透明
```

得到的结果如图 4-36（b）所示。



（a）非透明球体



（b）透明球体

图 4-36

4.2.5 立体可视化

除了前面介绍的常用网格图、表面图和等高线图外，MATLAB 还提供了一些立体可视函数用于绘制更为复杂的立体和向量对象。这些函数通常在三维空间中构建标量和向量的图形。由于这些函数构建的是立体而不是一个简单的表面，因此它们需要三维数组作为输入参数，其中三维数组的每一维分别代表一个坐标轴，三维数组中的点定义了坐标轴栅格和坐标轴上的坐标点。如果要绘制的函数是一个标量函数，则绘图函数需要 4 个三维数组，其中 3 个数组各代表一个坐标轴，第四个数组代表了这些坐标处的标量数据，这些数组通常记作 *X*、*Y*、*Z* 和 *V*。如果要绘制的函数是一个向量函数，则绘图函数需要 6 个三维数组，其中 3 个各表示一个坐标轴，3 个用来表示坐标点处的向量，这些数组通常记做 *X*、*Y*、*Z*、*U*、*V* 和 *W*。

要正确合理地使用 MATLAB 提供的立体和向量可视化函数，用户需要对与立体和向量有关的一些术语有所了解。比如，散度（Divergence）和旋度（Curl）用于描述向量过程，而等值面（Isosurfaces）和等值顶（Isocaps）则用于描述立体的视觉外观。如果用户要生成和处理比较复杂的立体对象，就需要参考相应的文献对这些术语进行深入了解。本书并不详细讲述这些术语的具体含义，而只是通过几个简单的例子讲述 MATLAB 中如何利用数据数组创建立体结构。关于这方面更详细的信息请读者参考相应的 MATLAB 在线帮助文档。

下面我们看一个利用标量函数构建立体图形的例子（例程 4.37）。首先，我们必须生成一个构建立体对象的坐标系。代码如下：

例程 4.37　构建立体对象坐标系

```
>> x=linspace(-3,3,13);
>> y=1:20;
>> z=-5:5;
>> [X,Y,Z]=meshgrid(x,y,z);
>> size(X)
```

结果如下：

```
ans =
     20    13    11
```

上面的代码演示了 meshgrid 函数在三维空间中的应用。其中，*X*、*Y*、*Z* 为定义栅格的 3 个三维数组。这 3 个数组分别是从 *x*、*y* 和 *z* 经过三维栅格扩展形成的。我们需要定义一个以这三个数组为自变量的标量函数 V，代码如下：

```
>> V=sqrt(X.^2+cos(Y).^2+Z.^2);
```

这样，利用标量函数 $v=f(x,y,z)$ 定义一个立体对象所需要的数据已全部给出。为了使该立体对象可视化，可以利用下面的代码查看该立体对象的一些截面。

```
>> slice(X,Y,Z,V,[0 3],[5 15],[-3 5])
>> xlabel('X_axis')
>> ylabel('Y_axis')
```

运行结果如图 4-37 所示。

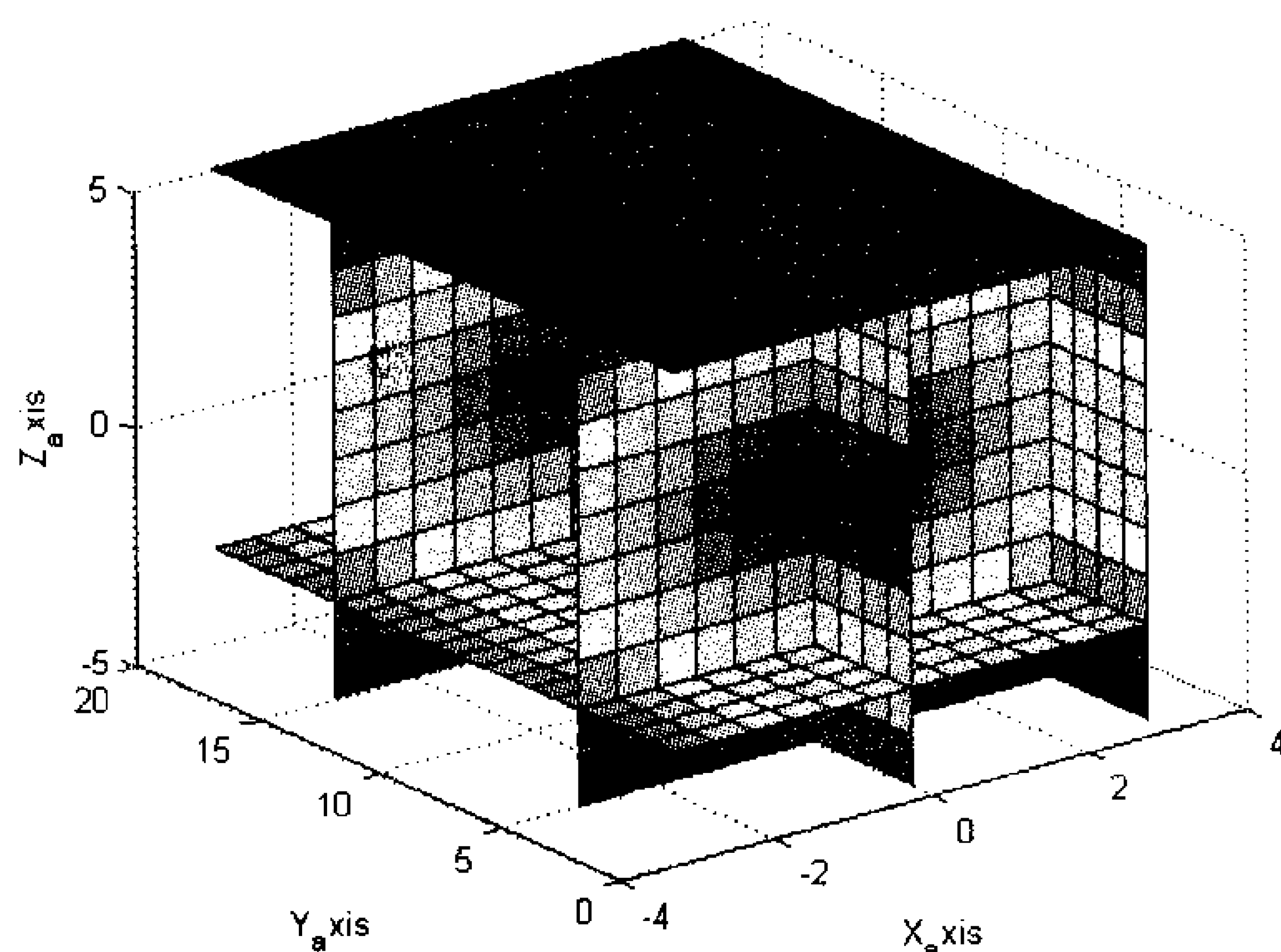


图 4-37　立体截面图

图 4-37 显示了立体图形在 $x=0$、$x=3$、$y=5$、$y=-15$、$z=-3$ 和 $z=5$ 所定义的平面上的截面。图中的颜色是根据截面上的 V 值来进行绘制的。

上图演示了立体图形的平面截面，在立体图形中，也可以显示立体图形的曲面截面。例如，下面的例子采用正弦函数来截取立体图形的截面，其中 *xs*、*ys* 和 *zs* 定义了一个截取立体图形的正弦截面。

例程 4.38　采用正弦函数来截取立体图形

```
>> [xs,ys]=meshgrid(x,y);
>> zs=sin(-xs+ys/2);
>> slice(X,Y,Z,V,xs,ys,zs)
>> xlabel('X_axis'),ylabel('Y_axis'),zlabel('Z_axis')
```

运行结果如图 4-38 所示。

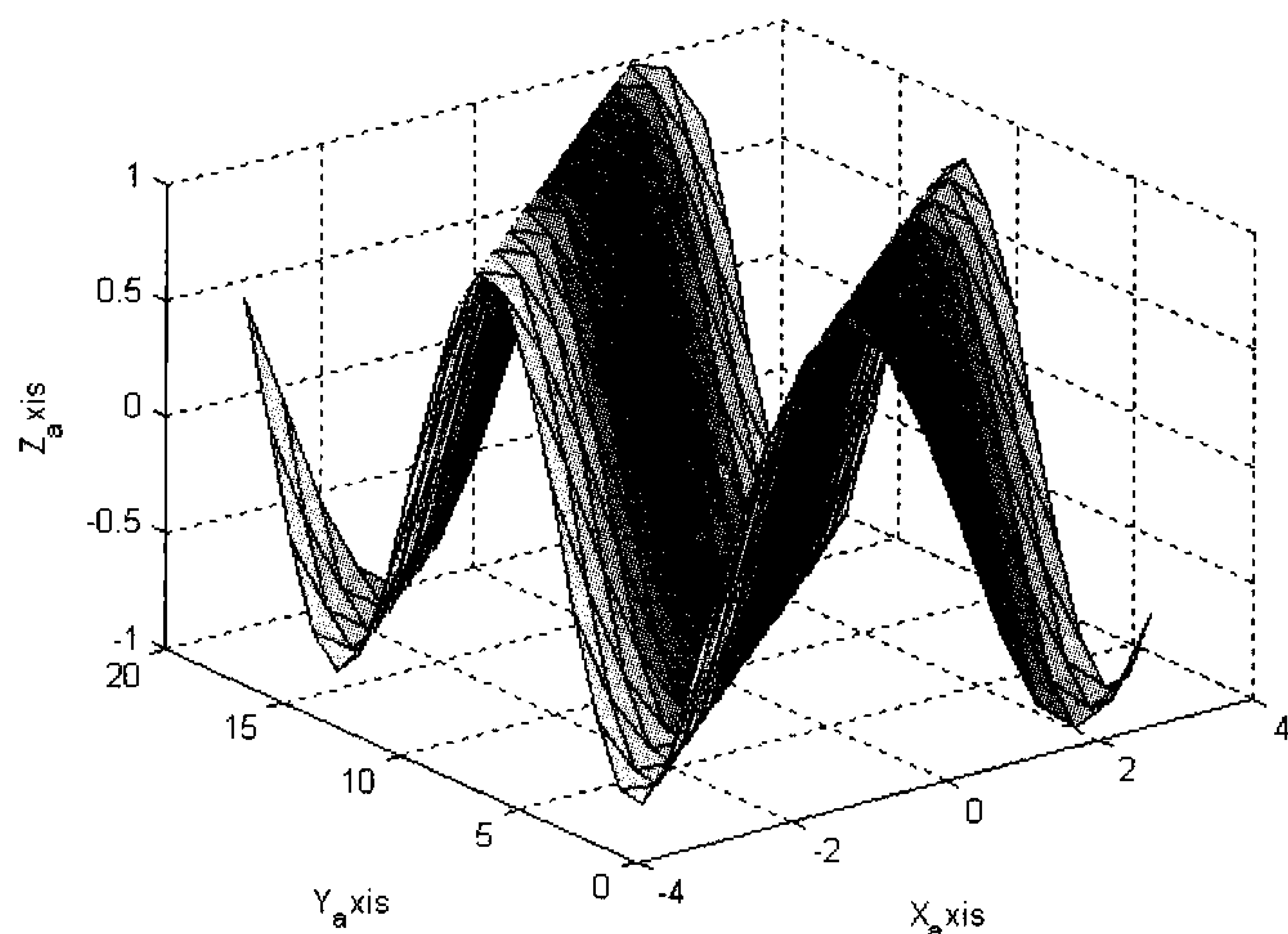


图 4-38　使用正弦曲面截取立体图形

除了截取平面以外，用户还可以用 contourslice 函数为截取的平面添加等高线，如下例所示。

例程 4.39　添加等高线

```
>> slice(X,Y,Z,V,[0 3],[5 15],[-3 5])
>> hold on
>> h=contourslice(X,Y,Z,V,3,[5 15],[]);
>> set(h,'EdgeColor','k','Linewidth',1.5)
>> xlabel('X_axis');ylabel('Y_axis');zlabel('Z_axis')
>> hold off
```

运行结果如图 4-39 所示。

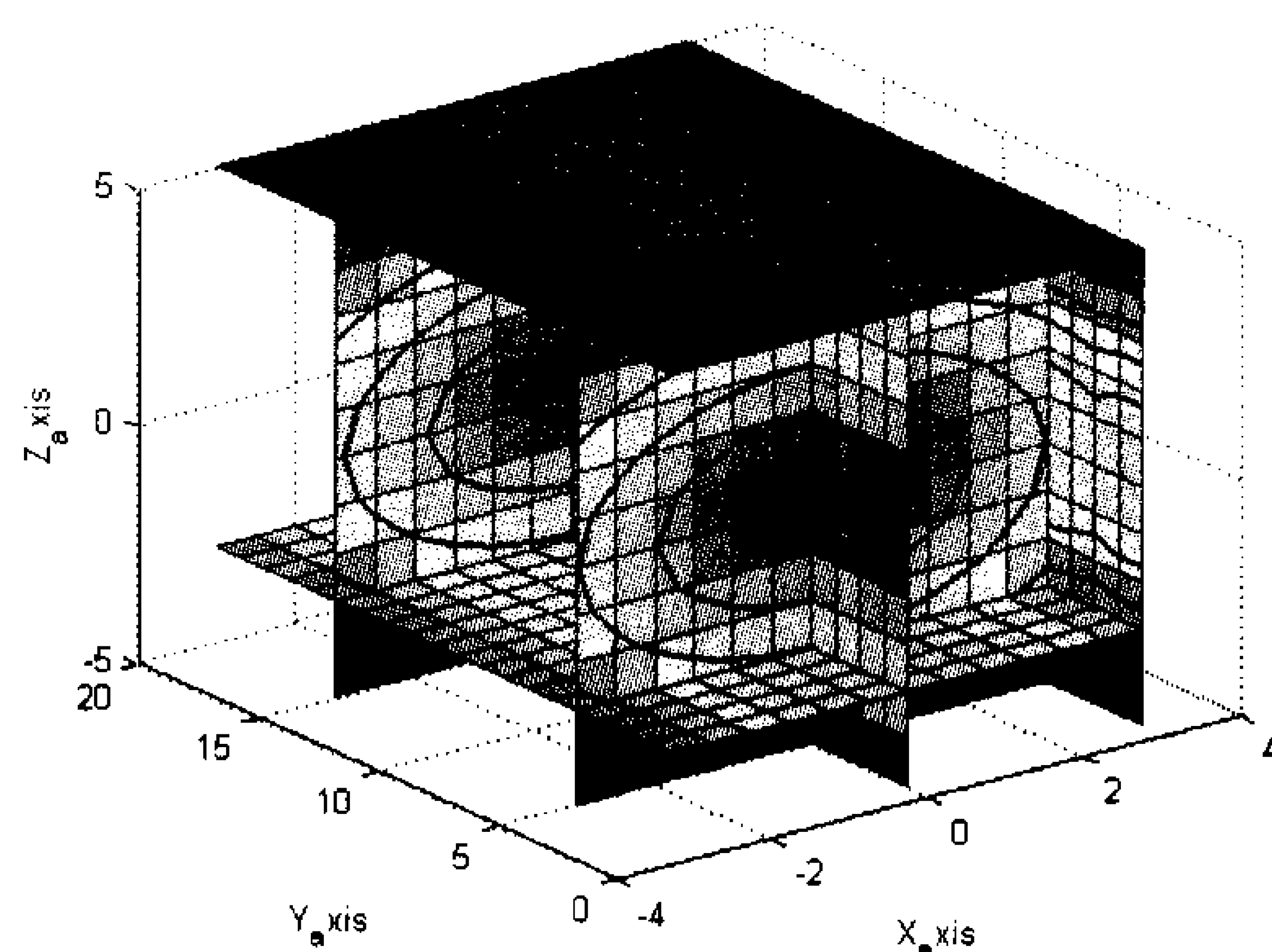


图 4-39　在立体图形的截面上绘制等高线

图 4-39 中，等高线被分别添加到了 x=3、y=5 和 y=15 的截面上，并利用句柄图形函数 set 将其颜色设置为黑色，宽度设置为 1.5 个像素点。

除了查看立体对象的截面之外，寻找使 V 等于某个特定值的表面（称为等值面）也十分常见。在 MATLAB 中，这一操作可以用 isosurface 函数来实现，该函数与 delaunay 函数

类似，由这些三角形构成的等值面。下面给出了一个绘制等值面的例子。

例程 4.40　绘制等值面

```
>> [X,Y,Z,V]=flow(13);
>> fv=isosurface(X,Y,Z,V,-2);
>> subplot(1,2,1)
>> p=patch(fv);
>> set(p,'FaceColor',[.5 .5 .5],'EdgeColor','Black');     %设置面属性
>> view(3),axis equal tight,grid on       %按比例显示图形，打开格网

>> subplot(1,2,2)
>> p=patch(shrinkfaces(fv,.3));
>> set(p,'FaceColor',[.5 .5 .5],'EdgeColor','Black');     %设置面属性
>> view(3),axis equal tight,grid on      %%按比例显示图形，打开格网
```

运行结果如图 4-40 所示。

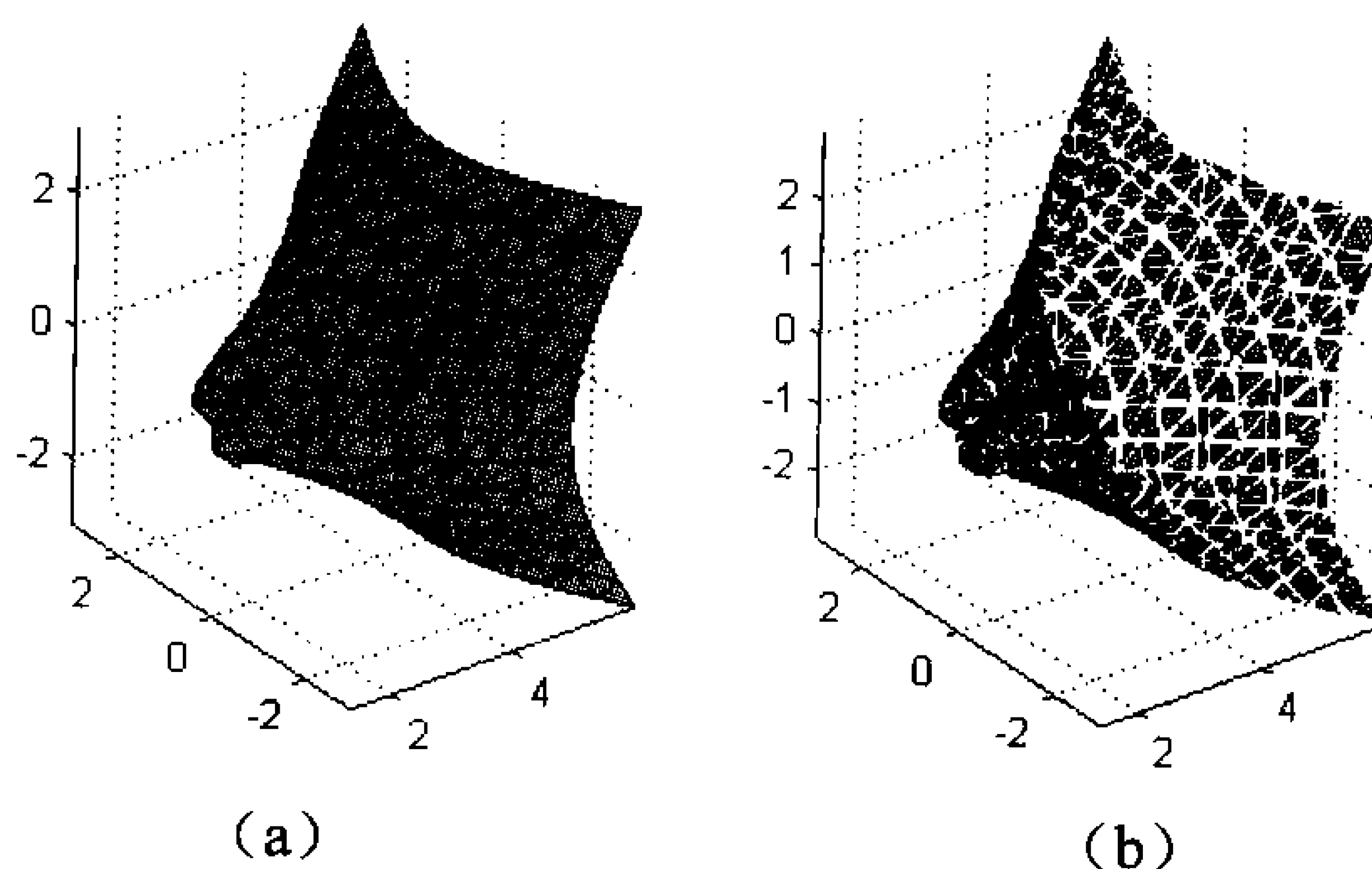


图 4-40　三维等值图

图 4-40（b）还展示了函数 shrinkfaces 的用法，顾名思义，该函数的功能为使表面收缩。

当我们显示立体图形仅仅是为了观察其大体结构时，就没有必要针对所有的数据点作图，因为数据点太多，会降低显示的速度。利用函数 reducevolume 和 reducepatch 则可以使用户在显示图形之前先删除一些数据或一些对图形显示影响很小的碎片，从而提高图形显示的效率。例程 4.41 演示了这两个函数的用法。

例程 4.41　reducevolume 和 reducepatch 函数用法

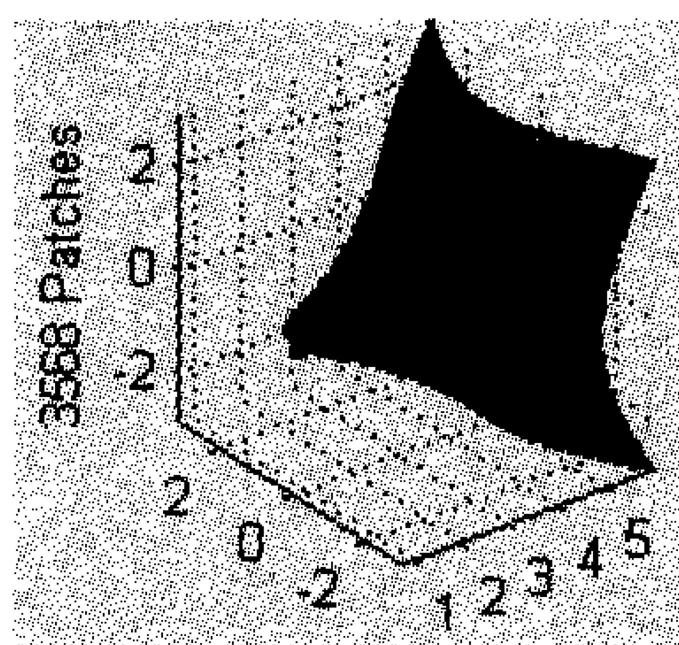
```
>> [X,Y,Z,V]=flow;
>> fv=isosurface(X,Y,Z,V,-2);
>> subplot(2,2,1)
>> p=patch(fv);
>> Np=size(get(p,'Faces'),1);
>> set(p,'FaceColor',[.5 .5 .5],'EdgeColor','Black');
>> view(3),axis equal tight,grid on
>> zlabel(sprintf('%d Patches',Np))

>> subplot(2,2,2)
>> [Xr,Yr,Zr,Vr]=reducevolume(X,Y,Z,V,[3 2 2]);
```

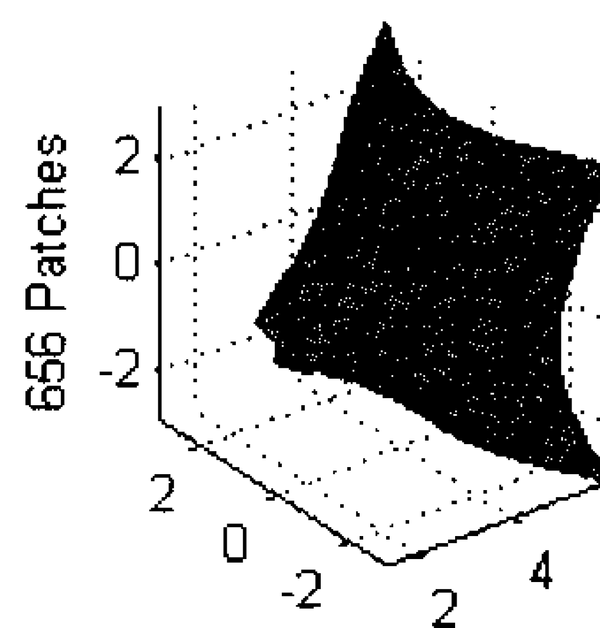
```
>> fvr=isosurface(Xr,Yr,Zr,Vr,-2);
>> p=patch(fvr);
>> Np=size(get(p,'Faces'),1);
>> set(p,'FaceColor',[.5 .5 .5],'EdgeColor','Black');
>> view(3),axis equal tight,grid on
>> zlabel(sprintf('%d Patches',Np))
>> subplot(2,2,3)
>> p=patch(fv);
>> set(p,'FaceColor',[.5 .5 .5],'EdgeColor','Black');
>> view(3),axis equal tight,grid on
>> reducepatch(p,.15)
>> Np=size(get(p,'Faces'),1);
>> zlabel(sprintf('%d Patches',Np))

>> subplot(2,2,4)
>> p=patch(fvr);
>> set(p,'FaceColor',[.5 .5 .5],'EdgeColor','Black');
>> view(3),axis equal tight,grid on
>> reducepatch(p,.15)
>> Np=size(get(p,'Faces'),1)
>> zlabel(sprintf('%d Patches',Np))
```

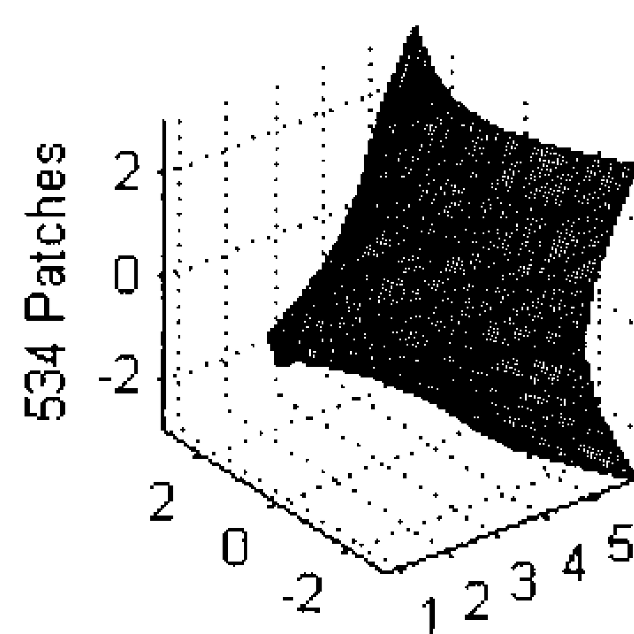
运行结果如图 4-41 所示。



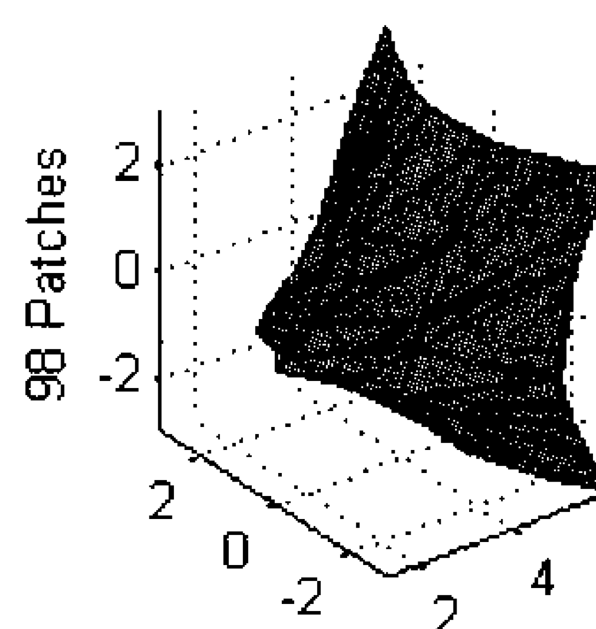
（a）



（b）



（c）



（d）

图 4-41 利用不同大小的碎片绘制三维表面

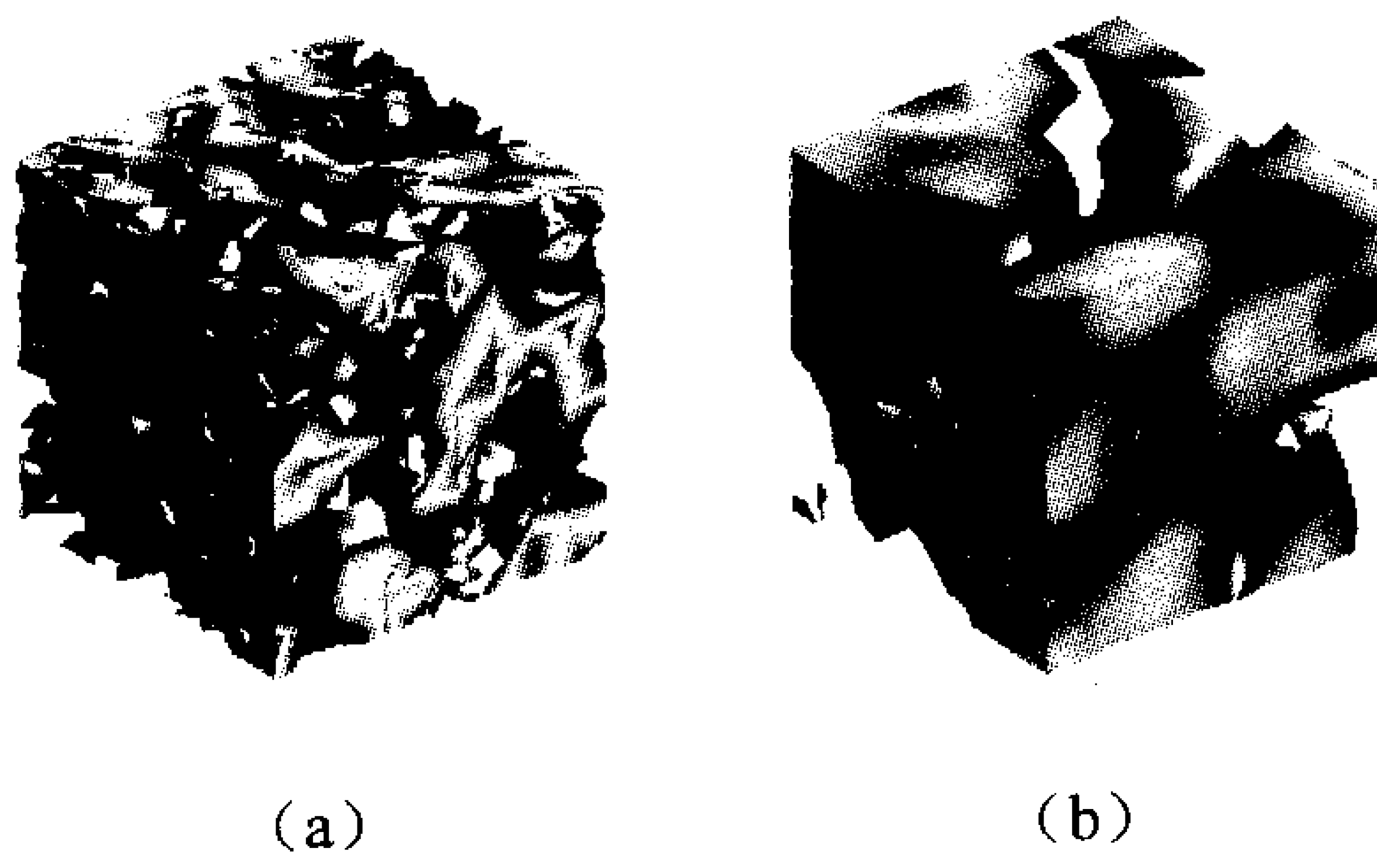
三维数据也可以通过用 smooth3 函数来过滤而实现其平滑化，如例程 4.42。

例程 4.42 smooth3 函数用法

```
>> data=rand(10,10,10);
>> datas=smooth3(data,'box',3);
>> subplot(1,2,1)
```

```
>> p=patch(isosurface(data,.5), 'FaceColor','Blue','EdgeColor','none');
>> patch(isocaps(data,.5),'FaceColor','interp','EdgeColor','none');
>> isonormals(data,p)
>> view(3);axis vis3d tight off
>> camlight;lighting phong
>> subplot(1,2,2)
>> p=patch(isosurface(datas,.5),'FaceColor','Blue','EdgeColor','none');
>> patch(isocaps(datas,.5),'FaceColor','interp','EdgeColor','none');
>> isonormals(datas,p)
>> view(3);axis vis3d tight off    %设置坐标轴比例因子相等
>> camlight;lighting phong    %生成摄像机函数并将其放在合适的位置
```

运行结果如图 4-42 所示。



（a）　　　　　　（b）

图 4-42　三维数据平滑

上边的例子展示了函数 isocaps 和 isonormals 的用法。函数 isocaps 生成块状图的外层表面。函数 isonormals 调整所画碎片的属性，使得所显示的图形有正确的光照效果。

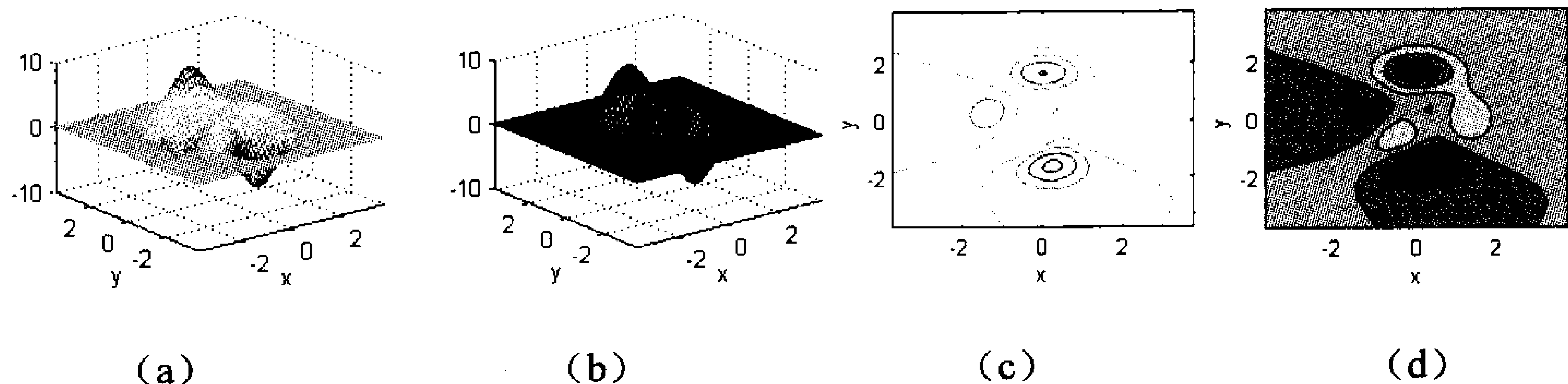
4.2.6　轻松绘制三维图形

当用户不想花费时间来显式地声明一个三维图形数据点的时候，MATLAB 还提供了函数 ezcontoru、ezcontour3、ezmesh、ezmeshc、ezplot3、ezsurf 和 ezsurfc。这些函数构建的图形类似于它们不带 ez 前缀的等价函数所构建的图形。但是，它们的输入参数是函数，这些函数是由字符串或符号数学对象，以及可选的图形所在的坐标轴所定义的。在这些函数内部，它们对数据进行计算，然后生成想要的图形，如例程 4.43 所示。

例程 4.43　快速绘图

```
>>fstr=['3*(1-x).^2.*exp(-(x.^2)-(y+1).^2)-10*(x/5-x.^3-y.^5).*exp(-x.^2-y.^2)-1/3*exp(-(x+1).^2-y.
^2)'];
>> subplot(2,2,1);ezmesh(fstr)
>> subplot(2,2,2);ezsurf(fstr)
>> subplot(2,2,3);ezcontour(fstr)
>> subplot(2,2,4);ezcontourf(fstr)
>> subplot(2,2,4);ezcontourf(fstr)
```

输出结果如图 4-43 所示。



(a) (b) (c) (d)

图 4-43 轻松绘图示例

4.3 图形色彩处理

MATLAB 提供了许多在二维和三维空间内显示信息的工具，但很多时候，一个简单的二维或者三维图形不能一次性显示出想要提供的信息。这时，颜色可以对图形提供一个附加的维数。本节的讨论从研究颜色映像开始，讲述如何创建、使用、显示和修改用户的颜色映像。

4.3.1 颜色映像原理

颜色映像就是把三基色——红色（R）、绿色（G）和蓝色（B）按照不同的比例组合起来，形成新的颜色。颜色映像的数据结构是若干行 3 列的矩阵，矩阵元素为 0～1 之间的数，这些数表示相应颜色的强度。颜色映像只在表面、补片和图像对象中使用。若干常用色的 RGB 值如表 4-32 所示。

表 4-32 若干常用色的 RGB 值

R	G	B	调和色	色符
0	0	1	蓝色	b
0	1	0	绿色	g
1	0	0	红色	r
0	1	1	青色	c
1	0	1	品红色	m
1	1	0	黄色	y
0	0	0	黑色	k
1	1	1	白色	w
0.5	0.5	0.5	灰色	
2/3	0	1	紫色	
1	0.5	0	橙色	
1	0.62	0.40	铜色	
0.49	1	0.83	宝石蓝色	

MATLAB 提供了几种典型的颜色映像，它们各侧重于不同的色调，这些颜色映像如表 4-33 所示。

表 4-33 几种典型色调的颜色映像

颜 色 映 像	颜 色 范 围
hsv	色彩饱和值：从红色开始，依次经过黄、绿、青、蓝、紫，最后再回到红色。这种颜色映像尤其适合周期函数
hot	从黑到红到黄到白
gray	线性灰度
bone	带一点蓝色调的灰度
copper	线性铜色调
pink	粉红、柔和的色调
white	白色
flag	交替的红色、白色、蓝色和黑色
lines	线性颜色
colorcube	增强的颜色立方
vga	Windows 的 16 位颜色映像
jet	hsv 的一种编写（以蓝色开始和结束）
prism	棱镜。交替的红色、橘黄色、黄色、绿色和天蓝色
cool	青和洋红的色调
autumn	红、黄色调
spring	洋红、黄色调
winter	蓝、绿色调
summer	绿、黄色调

MATLAB 的每个图形窗口只有一个色图，色图为 $m\times3$ 的矩阵。按默认方式，上面所列的各种颜色映像产生一个 64×3 的矩阵，指定了 64 种 RGB 颜色的描述。这些函数都接受一个参量来指定所产生矩阵的行数。例如，hot(m) 产生一个 $m\times3$ 的矩阵，它包含的 RGB 颜色的色值范围从黑经过红、橘红和黄到白。

4.3.2 颜色映像的应用

在图形表示过程中，颜色的运用能反映出许多其他的图形信息，所以颜色映像运用也是一个十分重要的环节。

1. 基本的着色技术

由于色彩在表现图形中非常重要，所以 MATLAB 特别重视色彩处理，而色图是 MATLAB 着色的基础。

语句 colormap(M) 将矩阵 *M* 当做当前图形窗口所用的颜色映像。例如，colormap(cool) 装入了一个有 64 个输入项的 cool 颜色映像。colormap default 装入了默认的颜色映像 hsv。

函数 colormap 的调用格式如下：

```
colormap(MAP)
```

该函数用于把当前图形的颜色映像设为 *MAP*，*MAP* 可以是 MATLAB 提供的颜色映像，如 “*jet*”，也可以自己定义颜色映像矩阵。需要注意的是，矩阵 ***MAP*** 的行数不限，但

必须为 3 列。

至此，我们一直在讲颜色映像，那么这些映像对应的到底是怎样的颜色呢？我们没有一个直观的认识，下面就来介绍能将颜色映像直观显示的函数。

2．颜色映像的直观显示

颜色映像的直观显示有如下几个知识点。

1）观察颜色映像矩阵元素

可以通过多个途径来显示一个颜色映像，其中一个方法就是观察颜色映像矩阵的元素。在 MATLAB 命令窗口输入如下命令：

```
>> hot(8)
```

执行结果如下：

```
ans =
    0.3333         0         0
    0.6667         0         0
    1.0000         0         0
    1.0000    0.3333         0
    1.0000    0.6667         0
    1.0000    1.0000         0
    1.0000    1.0000    0.5000
    1.0000    1.0000    1.0000
```

上面的数据显示出第 1 行是 1/3 红色，而最后一行是白色。

2）rgbplot 函数

函数 rgbplot 直接把颜色映像矩阵中的 3 列数分别用红、绿、蓝 3 种颜色画出来，例如：

```
>>rgbplot(hot)
```

输出结果如图 4-44 所示。

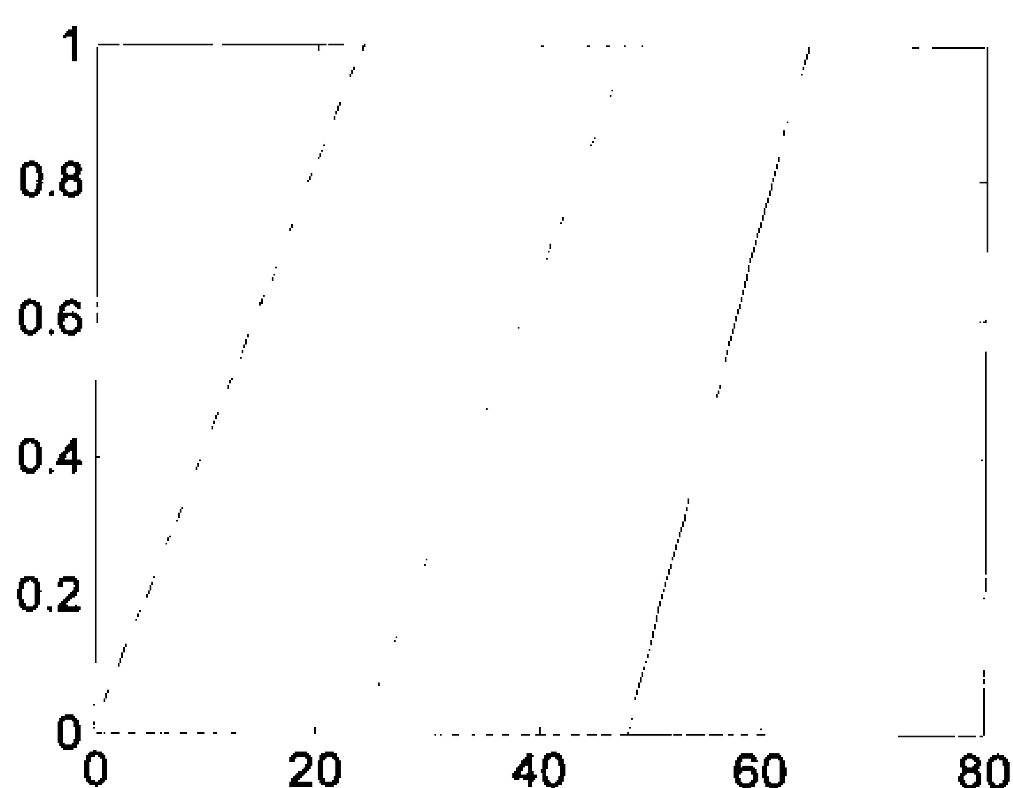


图 4-44　颜色映像 hot 的 RGB 曲线

上图绘制的就是颜色映像 hot 矩阵中的 RGB 数据，图中的红色曲线（左边）对应着矩阵的第 1 列；绿色曲线（中间）对应着矩阵的第 2 列；蓝色曲线（右边）对应着矩阵的第 3 列。从图中可以分析颜色的变换过程，图中从左到右表示颜色映像从开始到结束的变化，从下向上表示颜色的强度由小到大。开始 3 种颜色都是 0 或接近 0，因而是黑色；之后红色增强，而绿色和蓝色仍为 0，这一段为红色；红色到 1 后，绿色开始增强，蓝色仍然为 0，红色和绿色逐渐合成为黄色；绿色达到 1 后，蓝色逐渐增强，当蓝色最后也达到 1 后，3 种颜色合成白色。

3）pcolor 函数

函数 pcolor 用于绘制伪彩色图。所谓伪彩色是指绘图使用的色彩用于表示数据的大小，而不是自然的色彩。函数 pcolor 也可用来显示一个颜色映像，具体调用格式如下：

```
pcolor(c)
```

该函数的作用是将矩阵 C 作为颜色矩阵，利用着色原理在平面网格点(i,j)的右上角小区域内用 $C(i,j)$对应的色谱矩阵的颜色着色。绘制后的伪彩色图还可以用 shading 函数调整其颜色的平滑度。例如：

```
>>pcolor(cool(20))
```

其输出结果如图 4-45 所示。

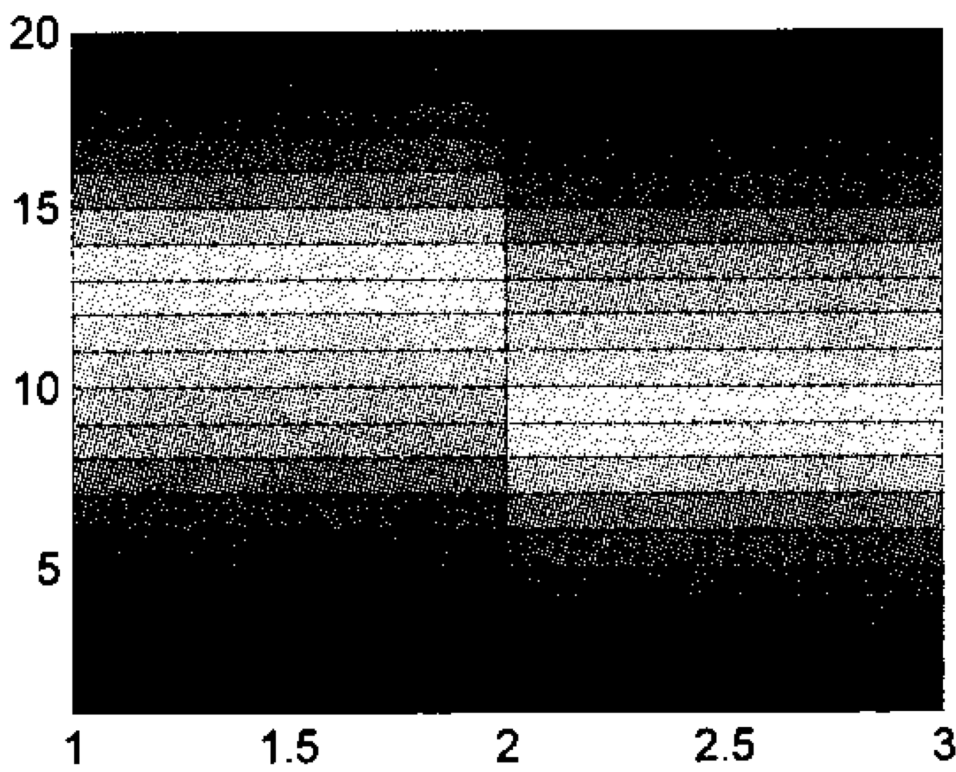


图 4-45　颜色映像 cool 的伪彩色图

4）colorbar 函数

函数 colorbar 在当前的图形窗口中增加水平或者垂直的颜色标尺以显示当前坐标轴的颜色映像。该函数的主要用法如表 4-34 所示。

表 4-34　colorbar 函数调用格式

调 用 格 式	说　明
colorbar	如果当前没有颜色条就加一个垂直的颜色条，或者更新现有的颜色条
colorbar('horiz')	在当前的图形下面放一个水平的颜色条
colorbar('vert')	在当前的图形右边放一个垂直的颜色条

3．颜色映像的建立和修改

颜色映像就是矩阵，意味着用户可以像其他数组那样对它们进行操作。MATLAB 提供了一系列的函数建立和修改颜色映像矩阵，如表 4-35 所示。

表 4-35　建立和修改颜色映像矩阵函数

函 数 名 称	描　述
brighten	通过调整一个给定的颜色映像来增加或者减少暗色的强度
caxis([cmin,cmax])	用于设置伪彩色的缩放比例。其中，“*cmin*”和“*cmax*”分别表示当前颜色映像中第一个颜色和最后一个颜色对应的数据大小

以下为这两个函数的应用示例。

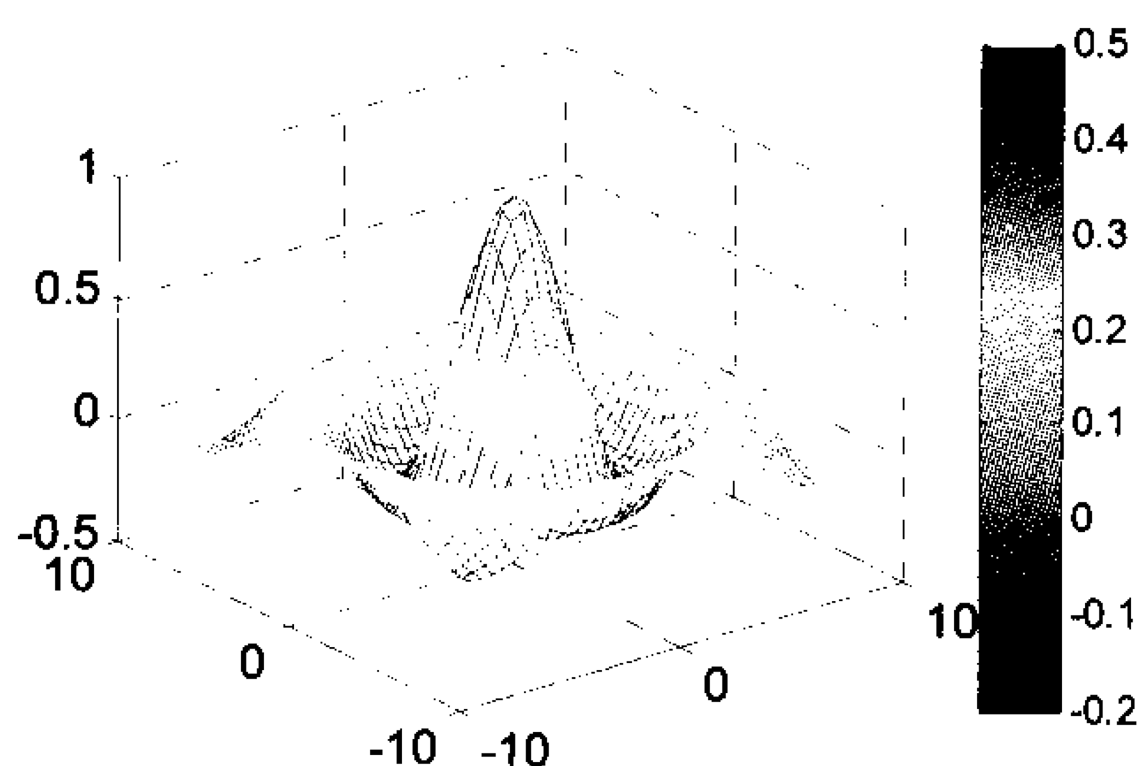
例程 4.44　caxis 函数使用示例

```
>> [X,Y]=meshgrid(-8:0.5:8);
>> R=sqrt(X.^2+Y.^2)+eps;
>> Z=sin(R)./R;
>> mesh(X,Y,Z)
>> caxis([-0.2,0.5])
>> colorbar('vert')
```

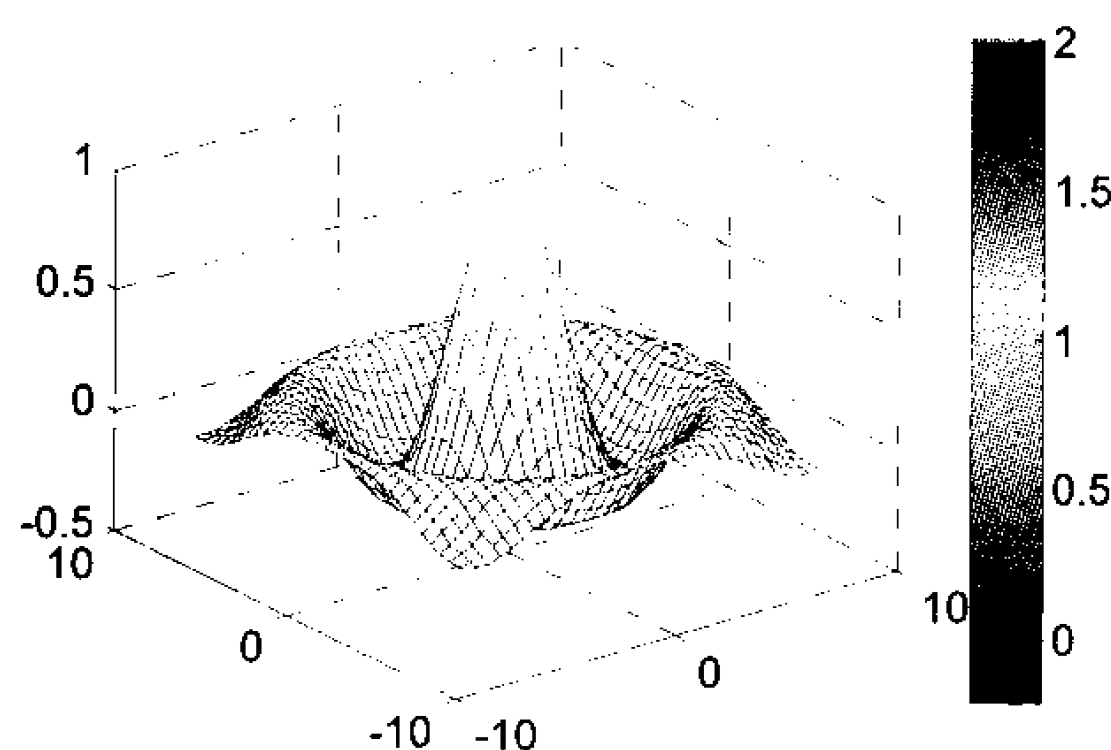
输出结果如图 4-46（a）所示。

```
>> caxis([-0.2,2])
>> colorbar('vert')
```

输出结果如图 4-46（b）所示。



（a）颜色映像的范围小于数据范围



（b）颜色映像的范围超出了数据范围

图　4-46

除此之外，用户还可以自定义颜色映像。

可以通过生成 $m\times3$ 的矩阵 mymap 来建立用户的颜色映像，并用 colormap（mymap）来安装它。颜色映像矩阵的每一个值都必须在 0 和 1 之间。如果企图用大于或小于 3 列的矩阵或者包含着比 0 小或者比 1 大的任意值，函数 colormap 会提示一个错误信息，然后退出。也可以组合颜色映像，其结果有时是不可预料的。只有当所有元素都在 0 与 1 之间时，才能保证结果是一个有效的颜色映像。

4.4　MATLAB 句柄式图形

在 MATLAB 中，句柄式图形是指一系列描述图形对象的表现形式和显示方式的底层图形特性函数的总称，每一个图形对象都有自己对应的句柄值，实质上前面各节中的图像、光照、颜色、线条、文本、表面等都是由句柄式图形创建的。句柄式图形的主要含义是允许用户依据句柄值来获取或设置对应图形对象的属性值，也就是通过图形对象的句柄值就可以一一设置对象的一些相关属性，可以将图形对象进行一次具有弹性的变化，使图形更加美观，满足用户不同的应用需求。

句柄式图形不但是对图形对象进行属性操作不可或缺的内容，也是进入 GUI 所要掌握的基本知识。通过本节的学习，读者也可以掌握 GUI 编程的基本知识。

本节深入讲述了图形对象之间的层次关系、常用图形对象的创建，以及如何获取图形

对象的句柄值，并通过句柄值与属性来控制对象。

4.4.1　图形对象和句柄式图形简介

在 MATLAB 中，所有的图形操作都是针对图形对象而言的，图形对象是图形系统中最基本、最底层的图元。如线条、图形、图片和图表等都是图形对象。它实际上进行着生成图形的工作，这些细节都隐藏在图形 M 文件的内部，但可以通过 MATLAB 函数来对它们进行操作。

详细来说，句柄式图形是基于这样一个概念，即一幅图的每一组成部分是一个对象，每一个对象都有一系列句柄和它相关，每一个对象有按需要可以改变的属性。它所支持的命令，可以直接创建线、文字、网格、面以及图形用户界面前面所介绍的高层图形命令（如 plot、mesh 等）。这些都是以句柄图形软件为基础写成的。因此，句柄式图形也被称为低层图形。句柄是存取图形对象唯一的规范识别符。不同对象的句柄不可能重复和混淆。

在句柄式图形体系中各个图形对象并不相等，它们之间有相对应的层次关系。这个关系可以用如图 4-47 所示的树结构层次表示。因此当某个父对象改变属性时，就会影响到该结构下层的所有子对象，如当前要改变窗口对象的位置，则线条与坐标轴对象也会跟着移动。一般来说，根对象的句柄值是 0，而窗口对象的句柄值通常设为整数，其他对象则用浮点小数值当做句柄值。

每建立一个图形对象的时候，MATLAB 就会自动建立该图形对象唯一的句柄值。如 fig=figure，则所建立的窗口中变量 fig 就是它的句柄值。

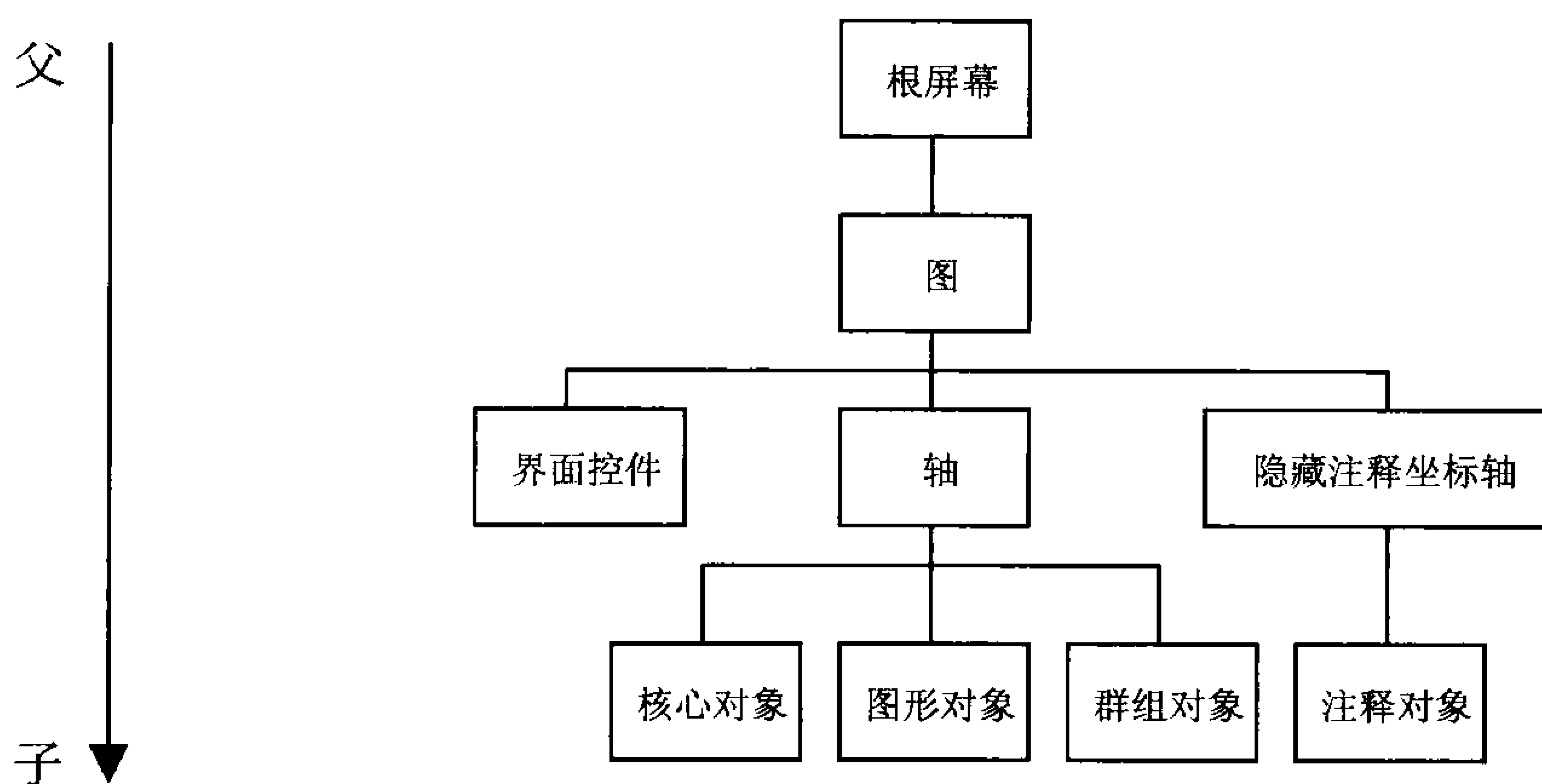


图 4-47　句柄式图形体现对象树结构

现对图 4-47 说明如下。

最上层为计算机屏幕，是所有对象的父对象，它所对应的子对象为 figure 绘图窗口，而图（父）所对应坐标轴、界面对象（包括界面菜单、界面控件等）、注释对象所存在的坐标轴隐藏层等为图的子对象，最后坐标轴（父）所对应为图形对象、核心对象、群组对象与注释对象等为坐标轴的子对象。再次说明阶层概念是如何运作的，举例来说，画一个线对象，MATLAB 需要建立坐标轴对象提供一个环境来进行线对象绘制；而一个坐标轴对象需要一个绘图窗口来显示坐标轴与它的子对象，所以基本上会有图、坐标轴与线 3 个对象存在。

坐标轴所对应的对象是用户所能感觉到的，本节讲述的图形对象属性操作就是对坐标轴所对应的对象的属性进行操作。句柄式图形对象按照层次关系一般可以分为四类：核心对象、群组对象、图形对象和注释对象。

1）核心对象（Core Objects）

核心的图形对象包含了一些基本的绘图函数，如 line、text 和 patch（多边形对象）；另外，比较特别的核心对象如 surface（用以组成矩形网格线顶点）、image 与 light 等，虽然它们不会显示出来，但会影响一些对象的颜色。如表 4-36 所示为核心图形对象的类型。

表 4-36　核心图形对象的类型

物　件	叙　述	物　件	叙　述
axes	在 figure 下的图面显示接口	patch	在 axes 中作多边形
image	在 axes 中的二维图面	rectangle	在 axes 中作矩形
light	在 axes 中的指示光源位置	surface	在 axes 中作三维绘图
line	在 axes 中作数据点的线连接	text	在 axes 中的字符

2）群组对象（Group Objects）

群组对象可以用来将坐标轴的子对象设置为一个群组，以便整个群组内对象属性的设置，如设置整个群组为可见或不可见，并且当选取一个群组对象后，底下的所有对象都会被选取。在 MATLAB 中有以下两种群组对象。

- hggroup：建立一个群组对象并且控制群组对象的可见性或可选择性来当做一个独立的对象时，就可以使用这个函数。
- hgtransform：将群组对象进行某些特征上的转换时（如将整个群组对象旋转与移动），就可以使用这个函数。

3）图形对象（Plot Objects）

一些可以用高级绘图方式来绘制图形的函数都可以用来建立图形对象（因为它们可以返回对应的对象句柄值，如 area 等）。在 MATLAB 中有些图形对象是由核心对象所组成的，因此可以通过核心对象的属性来控制这些图形对象的相关属性。如 fill 绘制的图形就是由 patch 对象所组成的，所以可以用与 patch 相关的属性来控制填充多边形图。图形对象的父对象可以为坐标轴或群组对象（如 hggroup 或 hgtransform），通过图 4-48 可以更方便地了解它们之间的关系，坐标轴对象为群组对象与图形对象的父对象，而群组对象也可以是图形对象的父对象。

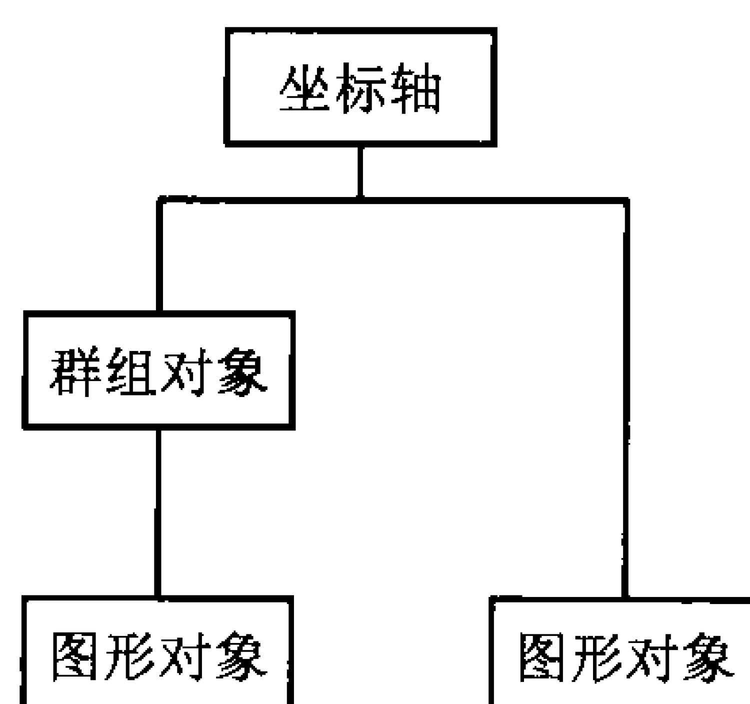


图 4-48　坐标轴与群组对象、图形对象的关系

如以下的示例是用来绘制 peaks 函数的等高线图形的，然后设置等高线的线条类型与

宽度。由于等高线是以 patch 对象（多边形对象）组成的，因此可以通过 patch 对象的 LineWidth 与 LineStyle 这两个属性来控制等高线。

例程 4.45　设置等高线属性

```
>> [X,Y,Z]=peaks;
>> [c,h]=contour(X,Y,Z);
>> set(h,'LineWidth',3,'LineStyle',':')      %设置线宽和线型
```

得到结果如图 4-49 所示。

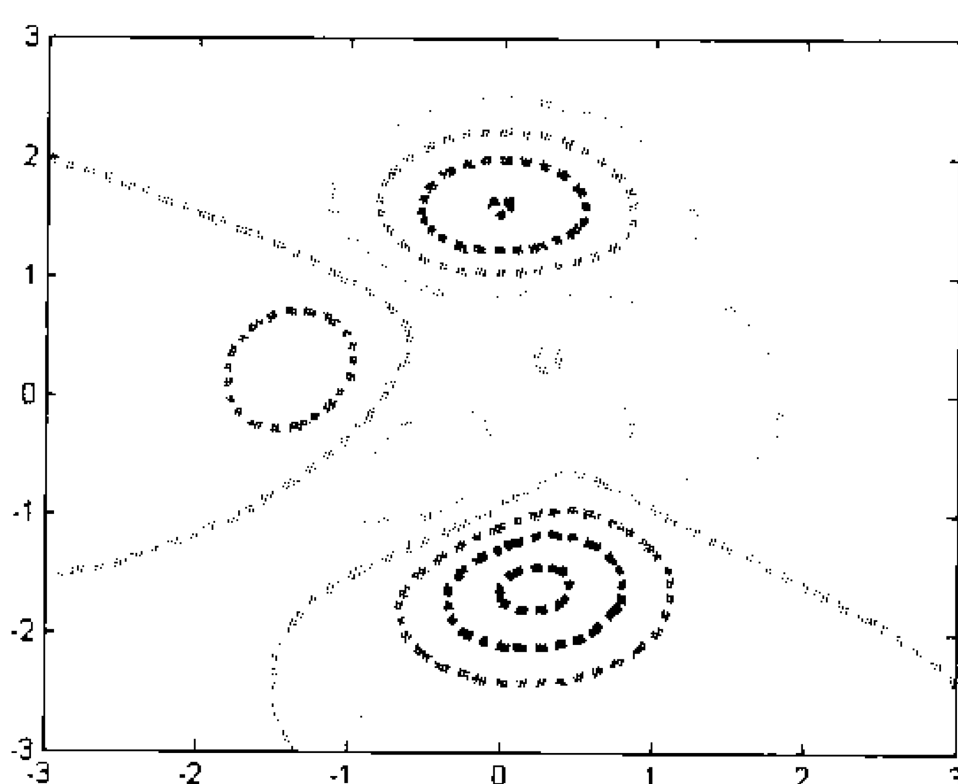


图 4-49　设置等高线属性

当观察等高线图内所包含的核心对象时，可以了解到等高线图是由 patch 对象的边缘线所组成的，同时也说明了我们可以利用 patch 对象的属性来设置等高线的属性，进而控制它的特征。

4）注释对象（Annotation Objects）

在 MATLAB 中，有 Arrow、Doublearrow、Ellipse、Line、Rectangle、Textarrow 和 Textbox 等几个注释对象。用户可以由绘图窗口的 Plot Edit 工具栏（必须通过执行【View】→【Plot Edit】命令来打开 Plot Edit 工具栏）或由菜单的“Insert”选项来建立注释对象。另一种方式是通过 annotation 函数来建立。注释对象建立在隐藏的坐标轴中，也就是说，它可以延伸宽与高在整个窗口中的显示，可以使用户通过正规化坐标（以左下角为原点（0，0），右上角为（1，1））的方式来定义注释对象在绘图窗口内的位置，如 TextBox。TextBox 是一个矩形的说明文字区域，它能够包含多行文字的输入，以填入适当的说明文字。单击【TextBox】按钮后，再通过单击鼠标左键来控制放置的位置即可。建立 TextBox 后，利用鼠标来改变其大小（直接拖曳 TextBox 对象的黑点）或位置（直接拖曳 TextBox 对象），并且直接在 TextBox 对象上双击鼠标左键就可以编辑该内容了，同样，在该对象上单击鼠标右键可以直接控制简易的属性，如 TextColor、BackgroundColar 和 EdgeColor 等，或选取“Properties”选项，就可以打开该对象的“Property Editor”对话框了。

所有的注释对象都会显示在一个覆盖于整个绘图窗口的重叠坐标轴中，并且这个坐标轴是隐藏的，因此在这一层重叠的坐标轴中，只会显示注释对象，也就是说，无法控制它的父对象或设置坐标轴的任何属性，但因为这个隐藏的坐标轴是与绘图窗口一样大的，因此可以将注释对象在绘图窗口中任意的位置显示，而不仅限于坐标轴内。当然，用户也可以通过 line、text 和 rectangle 这些函数在坐标轴的资料坐标内建立线、文字、矩形与椭圆，但是这些对象是不会放置在注释的坐标轴层中的（因为它们并非注释对象），并且它们必须存在于父对象（坐标轴）内，因此它们可以设置父对象的属性。以下是一个建立矩形注释

对象的范例，并且通过颜色的设置使子图在显示上更为明显。

例程 4.46　建立矩形注释对象

```
>> x=-2*pi:pi/12:2*pi;
>> y=x.^2;
>> subplot(2,2,1:2);
>> plot(x,y)
>> h1=subplot(2,2,3);
>> y=x.^4;
>> plot(x,y)
>> h2=subplot(2,2,4);
>> y=x.^5;
>> plot(x,y)
```

以下通过坐标轴的 Position 与 TightInset 属性来决定矩形注释对象的大小与位置，其中由于必须将封闭坐标轴、卷展栏、刻度与标题所占的区域大小都考虑进去，因此必须结合 Position 与 TightInset 这两个属性使用。如下所示，由于必须在子图 3 与子图 4 上加入矩形注释对象，因此必须先求得以下几个尺寸。

```
>> p1=get(h1,'Position');    %获得子图 3 坐标轴的大小与位置
>> t1=get(h1,'TightInset');
>> p2=get(h2,'Position');  %获得子图 4 坐标轴的大小与位置
>> t2=get(h2,'TightInset');
>> x1=p1(1)-t1(1);    %扣除卷展栏、刻度与标题所占的空间，求得 x1、x2、y1 与 y2
>> y1=p1(2)-t1(2);
>> x2=p2(1)-t2(1);
>> y2=p2(2)-t2(2);
>> w=x2-x1+t1(1)+p2(3)+t2(3);    %计算矩形注释对象的宽与高
>> h=p2(4)+t2(2)+2(4);
>> h=p2(4)+t2(2)+t2(4);
```

以宽 *w* 来说，*x*2-*x*1 就是子图 3 坐标轴未包含左方卷展栏的距离加上子图 3 坐标轴与子图 4 坐标轴之间的空白区域的距离，*t*1(1)为子图 3 坐标轴左方包含卷展栏的距离，*p*2(3)为子图 4 坐标轴的宽，*t*2(3)为子图 4 坐标轴右方包含卷展栏的距离。在子图 3 与子图 4 上建立矩形注释对象，并且设置该矩形为红色而且具有实线边框。

得到的结果如图 4-50 所示。

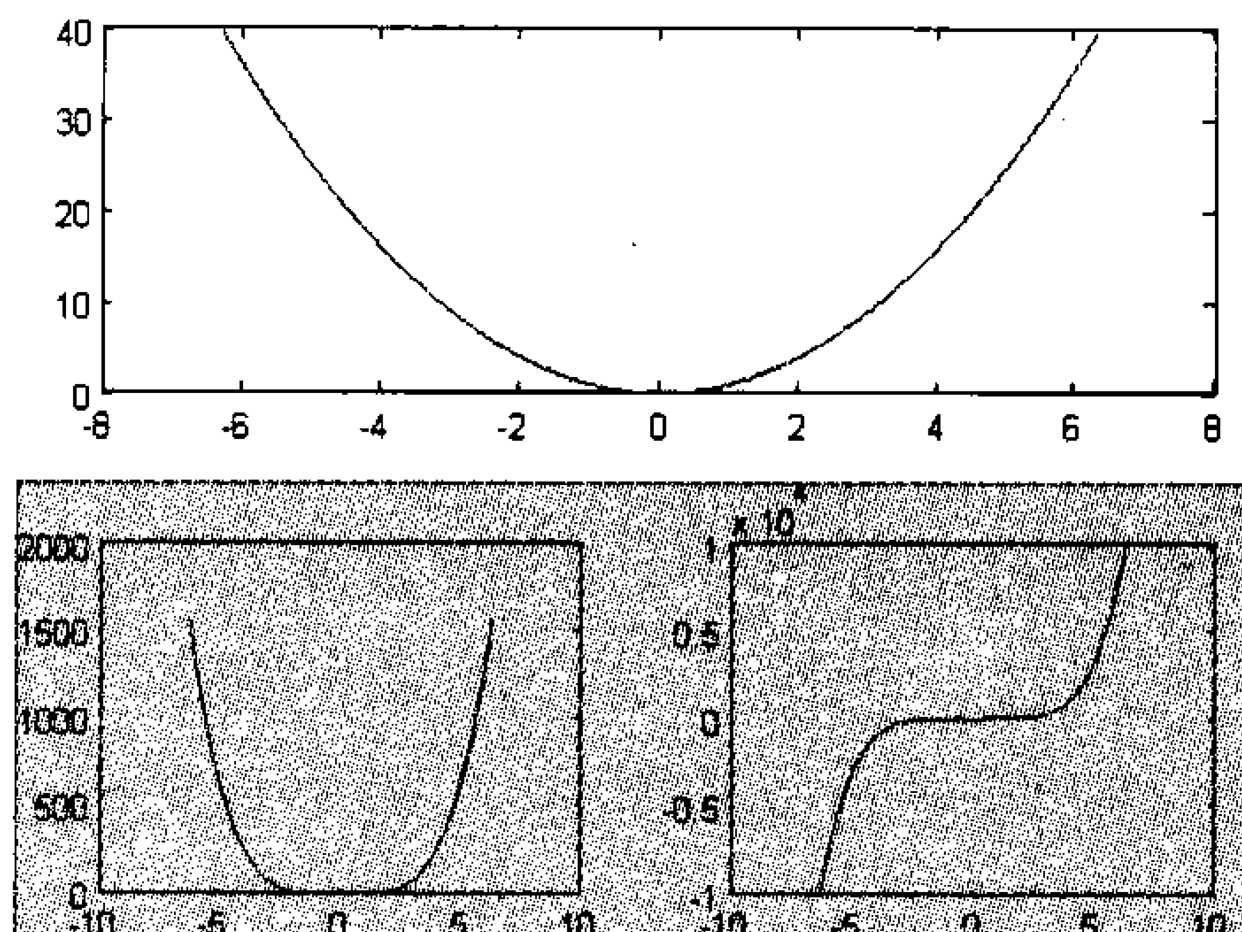


图 4-50　建立矩形注释对象示例

4.4.2 常用图形对象创建及其属性介绍

前面介绍了图形对象的基本概念，接下来介绍这些对象的创建方法。通过这些对象的内涵和功能来全面了解一下它们的基本属性。实际上，在 MATLAB 中，所有图形对象在调用对应函数的时候就自动创建了。本节内容请读者结合 MATLAB help 文档学习。

在 MATLAB 中，所有图形对象都由与之同名的函数创建。对象创建函数和含义如表 4-37 所示。

表 4-37　图形对象

对　象	父　类	描　述
figure	root	在 root 屏幕上分割窗口来显示图形。所有绘图函数在没有当前图形窗口时，都会自动创建一个 figure 对象
axes	figure	轴对象在窗口中定义一个图形区域。可以用来描述子对象的位置和方向
uicontrol	figure	用户界面控制。通过事件触发机制来执行回调函数
uicontextmenu	figure	快捷菜单
uimenu	figure	菜单。以弹出窗的形式触发事件执行回调函数
image	axes	包含数组或色图
line	axes	MATLAB 最原始的绘图线函数
patch	axes	补片对象
rectangle	axes	矩形对象
surface	axes	实线或内插颜色来绘制面
text	axes	在 axes 对象上进行标注
light	axes	在图形上定义光源

接下来我们结合一些示例介绍这些图形对象的创建及其属性操作。限于篇幅，在这里我们只对 figure 对象的属性进行列举，其他对象的属性与 figure 属性类似，具体请参阅 MATLAB help 帮助文档。

1．figure 及其相关命令

figure 对象是 MATLAB 系统中包括 GUI 设计编辑窗在内所有显示的窗口。在系统运行极限条件下，用户可以创建任意多个 figure 窗口。所有 figure 对象的父对象都是 root 对象，而其他所有 MATLAB 对象都是 figure 对象的子对象。figure 函数可以建立一个新的 figure 对象。其调用格式如表 4-38 所示。

表 4-38　figure 函数调用格式

调 用 格 式	说　明
figure	使用默认的属性值来建立一个新的 figure 对象
figure('ProName','ProVal',...)	使用指定的属性名称的属性值来建立一个新的 figure 对象
figure(h)	表示打开一个句柄为 *h* 的新的 figure 窗口
h = figure(...)	返回窗口的句柄属性值向量

figure(h)的使用方式会有以下 3 种情况发生。

- 如果 h 为一个已经存在的 figure 句柄值，则 figure(h)会使 figure 识别 h 为当前的 figure 对象，使它为可见的，并且在屏幕上将其显示到所有图形之前。注意，当前的 figure 为图形输出的地方，也因为它为当前对象，所以可以直接控制它的属性。
- 如果 h 不是已经存在的 figure 句柄值，而是一个整数，则 figure 函数会产生一个新的 figue 窗口，同时把该 figure 窗口的句柄值设为 h。
- 如果 h 不是一个已经存在的 figure 窗口句柄值，也不是一个整数，则返回一个错误信息。

figure 对象属性如表 4-39 所示。

表 4-39　figure 对象的属性

属 性 值	描　述
figure 位置	
Position	窗口的大小和位置，四元向量，默认值由显示器分辨率决定
Units	显示的度量单位。值为 pixels（像素，默认）、normalized（归一化坐标）、inches（英寸）、centimeters（cm）、points（打印机的点，等于 0.353mm）、characters（字符）。Guide 设计默认值为 characters
指定类型与外在显示	
Color	图形窗口背景颜色
MenuBar	控制 MATLAB 菜单在图形窗口的顶部显示。值可以为“figure”（显示菜单，默认值）或者“none”（隐藏菜单）
Name	图形窗口标题，默认为空字符串
NumberTitle	标题栏中是否显示“figure No.n”，其中 n 为图形窗口的编号。值为“on”（默认）或“off”
Resize	指定图形窗口是否可以通过鼠标改变大小。值为“on”（默认）或“off”
SelectionHighlight	当图形窗口被选中时，是否突出显示。值为“on”（默认）或“off”
Visible	图形窗口是否可见。值为“on”（默认）或“off”
WindowStyle	指定窗口为标准窗口还是典型窗口。值为 normal（标准窗口、默认）或 modal（典型窗口）
控制色图	
Colormap	图形窗口的色图。值为 $m\times3$ 阶的 RGB 颜色矩阵，默认 jet 色图
Dithermap	用于真彩色数据以伪彩色显示的色图。值为 $m\times3$ 阶的 RGB 颜色矩阵
DithermapMode	是否使用系统生成的抖动色图。值为 auto、manual（默认）
FixedColors	非色图颜色。只读，值为 $m\times3$ 阶的 RGB 颜色矩阵
MinColormap	系统颜色表中最少的颜色数，值为一标量（默认 64）
ShareColors	是否允许 MATLAB 共享系统颜色表中的颜色。值为“on”（默认）或“off”
定义透明度	
Alphamap	图形窗口的透明度色图。值为 $m\times1$ 向量，每一分量在[0 1]之间，默认 64×1 向量
指定渲染模式	
BackingStore	打开或关闭屏幕像素缓冲区。值为“on”（默认）或“off”
DoubleBuffer	对于简单的动画渲染是否使用快速缓冲。值为“on”（默认）或“off”
Renderer	用于屏幕和图片的渲染模式。值为 painters、zbuffer、openGL（默认由系统选择）



（续表）

属 性 值	描 述
	图形窗口的一般信息
Children	显示在图形窗口中的任意对象句柄，值为句柄向量
Filename	GUI 设计的 fig 文件名
Parent	父对象。值总为 0
Selected	是否显示窗口选中状态。值为“on”（默认）或“off”
Tag	用户定义的对象标识。值为任意有效字符串
Type	对象类型。只读，总是“figure”
UserData	用户定义的数组
RendererMode	默认的或用户指定的渲染程序。值为“auto”（默认）或“manual”
	当前状态信息
CurrentAxes	在图形窗口中的当前坐标轴的句柄
CurrentCharacter	窗口中输入的最后一个字符。值为一个字符
CurrentObject	当前图形窗口中的对象句柄
CurrentPoint	图形窗口中最后单击的按钮的位置。二元向量
SelectionType	鼠标选择类型。值为“normal”（鼠标左键，默认）、“extended”（【shift】+鼠标左键或鼠标左右两键同时）、“alt”（【Ctrl】＋鼠标左键或鼠标右键）、“open”（鼠标任意键）
	回调函数执行
BusyAction	指定如何处理回调函数。值为“cancel”（当存在一个事件在执行时，新的事件自动释放）或“queue”（以队列的方式执行事件，默认）
ButtondownFcn	当对象空闲时，单击鼠标调用的回调函数。默认为空字符串
CloseRequestFcn	关闭图形窗口时调用的回调函数。默认 closereq
CreateFcn	窗口创建时调用的回调函数。默认为空字符串
DeleteFcn	删除图形窗口时调用的回调函数。默认为空字符串
Interruptible	定义回调函数是否可以中断。值为“on”（默认）或“off”
KeyPressFcn	在图形窗口中按下键盘键时调用的回调函数。默认为空字符串
ResizeFcn	当图形改变大小时调用的回调函数。默认为空字符串
UIContextMenu	与图形对象相关的 uicontextmenu 对象句柄
WindowButtonDdownFcn	当图形窗口中按下鼠标键时调用的回调函数。默认为空字符串
WindowButtonMotionFcn	当鼠标移进图形窗口时调用的回调函数。默认为空字符串
WindwoButtonUpFcn	当在图形窗口中松开鼠标时调用的回调函数。默认为空字符串
	对象访问控制
IntegerHandle	指定使用整数或非整数图形对象。值为“on”（默认）或“off”
HandleVisibility	图形对象句柄是否可见。值为“on”（默认）、“callback”或“off”
HitTest	定义图形窗口是否能变成当前对象（参见属性 CurrentObject）。值为“on”（默认）或“off”
NextPlot	如何添加图形。值为“add”（用当前图形窗口显示图形，默认）、“replace”（复位图形窗口对象，删除子对象）或“replacechildren”（删除子对象）
	定义鼠标指针
Pointer	定义指针形状。值为 crosshair、arrow（默认）、topr、watch、topl、botl、botr、circle、cross、fleurleft、right、top、fullcrosshair、bottom、ibeam、custom
PointerShapeCData	自定义鼠标外形。值为 16×16 阶矩阵，默认为将鼠标设置为“custom”且可见
PointerShapeHotSpot	鼠标回调的点。值为二元向量。对应参数 PointerShapeCData

（续表）

属 性 值	描 述
打印参数设置	
InvertHardcopy	互换打印的颜色。值为“on”（不改变颜色，默认）或“off”
PaperOrientation	打印的方向。值为“portait”（垂直方向，默认）或“landscape”（水平方向）
PaperPosition	定义打印图形窗口的位置。值为[left,bottom,width,height]
PaperPositionMode	图形输出在纸张上的位置是手动还是自动给出。值为“manual”（MATLAB 会使用 PaperPosition，默认）或“auto”（输出的位置就和在屏幕上看到的一样）
PaperSize	规定纸的大小。值为二元向量
PaperType	打印图形纸张的类型，值为“usletter”（默认）、“uslegal”、“A0”、“A1”、“A2”“A3”、“A4”、“A5”、“B0”、“B1”、“B2”、“B3”、“B4”、“B5”、“arch-A”、“arch-B”、“arch-C”、“arch-D”、“arch-E”、“A”、“B”、“C”、“D”、“E”或“tablodi”
PaperUnits	显示的度量单位。值为“normalized”（归一化坐标）、inches（英寸，默认）、centimeters（cm）、points（打印机的点，等于 0.353mm）

例程 4.47 figure 对象示例

```
%创建一个默认属性值的图形窗口
figure
%创建一个自定义属性值窗口
%定义窗口侧面与底部边界宽度为 5 个像素，而顶部的边界宽度为 30 个像素
Bdwidth=5;topbdwidth=30;
figure('color',[1 1 1],'MenuBar','figure','Name','Figure 对象示例','NumberTitle','off')
```

得到的结果如图 4-51 所示。

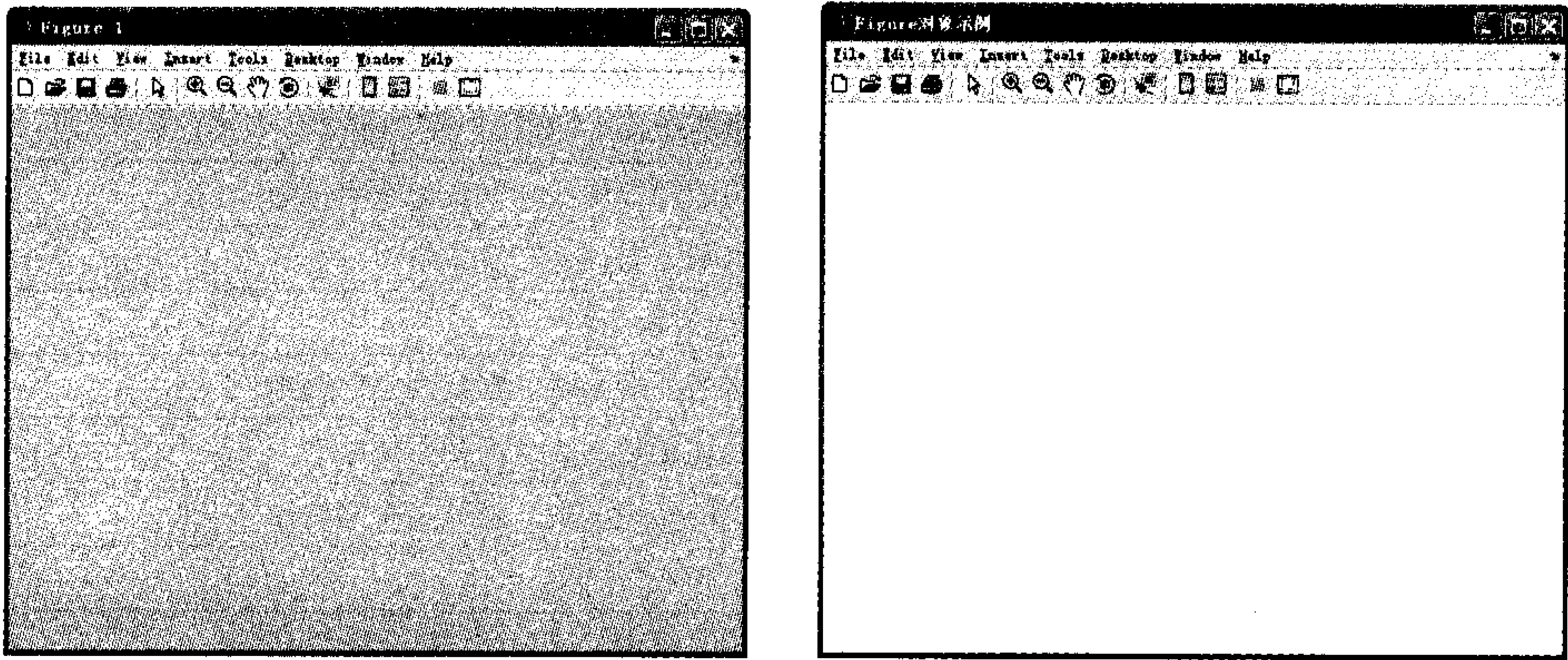


图 4-51 figure 创建图形示例

2．axes 及其相关命令

轴（Axes）对象在图形窗口中定义一个画图区域。如果在 MATLAB 系统运行中没有轴对象，则任意一个绘图函数都在绘图事件前自动创建一个轴对象（没有 figure 对象也自动创建），并把它设置为“当前”轴坐标。以后的绘图函数没有特别指定，都以它为绘图区域。图形函数 image、line、patch、surface 和 text 等命令也把它作为输出图形区域。轴对象由 axes 函数来创建。该函数调用格式如表 4-40 所示。

表 4-40　axes 函数调用格式

调用格式	说　　明
axes	使用默认的属性值来建立一个新的 axes 对象
axes('ProName','ProVal',...)	使用指定的属性名称的属性值来建立一个新的 axes 对象
axes(h)	表示打开一个句柄为 *h* 的新的 axes 窗口
h = axes(...)	返回坐标轴的句柄属性值向量

例程 4.48 为 axes 函数应用示例。

例程 4.48　axes 对象示例

```
%左下的坐标轴
axes_handles(1)=axes('Position',[0.1 0.05 0.2 0.2]);
%以下绘图语句所绘制的图形会在左下的坐标轴中显示
k=1:30;
[B,XY]=bucky;
gplot(B(k,k),XY(k,:),'-*')
axis square

%右下的坐标轴
axes_handles(2)=axes('Position',[0.7 0.05 0.2 0.2]);
%以下绘图语句所绘制的图形会在右下的坐标轴中显示
x=rand(1,50);
y=rand(1,50);
z=peaks(6*x-3,6*x-3);
tri=delaunay(x,y);
trisurf(tri,x,y,z);

%右上的坐标轴
axes_handles(3)=axes('Position',[0.7 0.75 0.2 0.2]);
%以下绘图语句所绘制的图形会在右上的坐标轴中显示
[X,Y,Z]=peaks(30);
surfc(X,Y,Z)
colormap hsv
axis([-3 3 -3 3 -10 5]);

%左上的坐标轴
axes_handles(4)=axes('Position',[0.1 0.75 0.2 0.2]);
%以下绘土语句所绘制的图形会在左上的坐标轴中显示
x=-pi:.1:pi;
y=sin(x);
plot(x,y);
set(gca,'XTick',-pi:pi/2:pi);
set(gca,'XTickLabel',{'-pi','-pi/2','0','pi/2','pi'})

%中间的坐标轴
axes_handles(5)=axes('Position',[0.3 0.3 0.4 0.4]);
%以下绘土语句所绘制的图形会在中间的坐标轴中显示
t=0:pi/50:10*pi;
plot3(sin(t),cos(t),t);
```

```
grid on
axis square
```

得到的结果如图 4-52 所示。

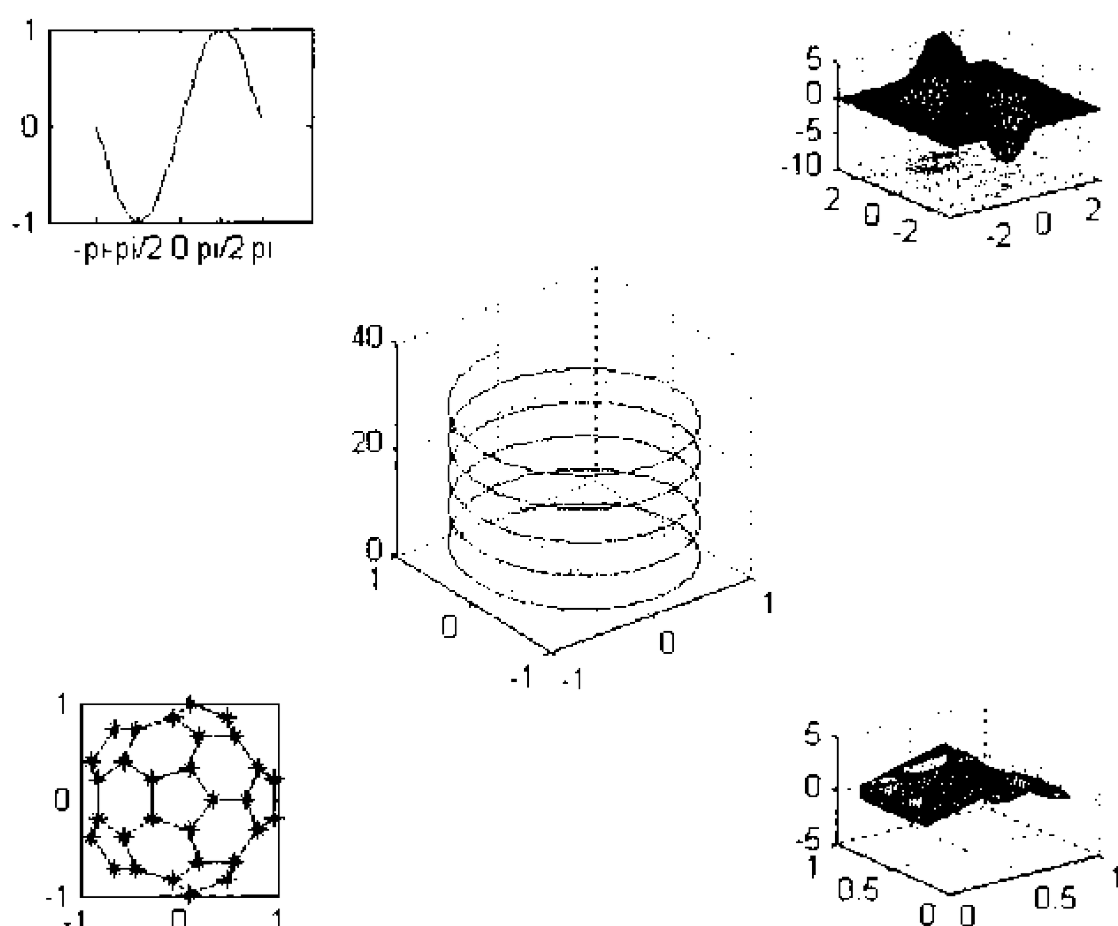


图 4-52　axes 对象示例

3．image 及其相关命令

MATLAB 中也可以显示图像。MATLAB 中的图像是由矩阵来定义的，矩阵的元素相对应于图像中的点，元素的值相对应于点的颜色，所以 MATLAB 中的图像是“位图”类型。根据数据矩阵元素代表的意思不同，MATLAB 中的 image 对象有 3 种基本的类型，即索引图像、灰度图像（包含二值图像）和真彩色图像。

image 对象由 image 函数来进行创建，该函数的用法将在图像显示技术中详细介绍，请读者参阅。例程 4.49 为 image 函数应用示例。

例程 4.49　通过 image 函数来控制图像的显示

```
>> load earth      %载入地球图像 MAT 文件
>> image(X,'CDataMapping','scaled');
>>colormap(map)
>> axis square     %坐标轴长与宽等距
```

首先加载地球的图像 MAT 文件，该 MAT 文件中的 X 为图像的矩阵数据，map 为颜色映像值，用以控制该图像的颜色。例程 4.49 执行结果如图 4-53 所示。

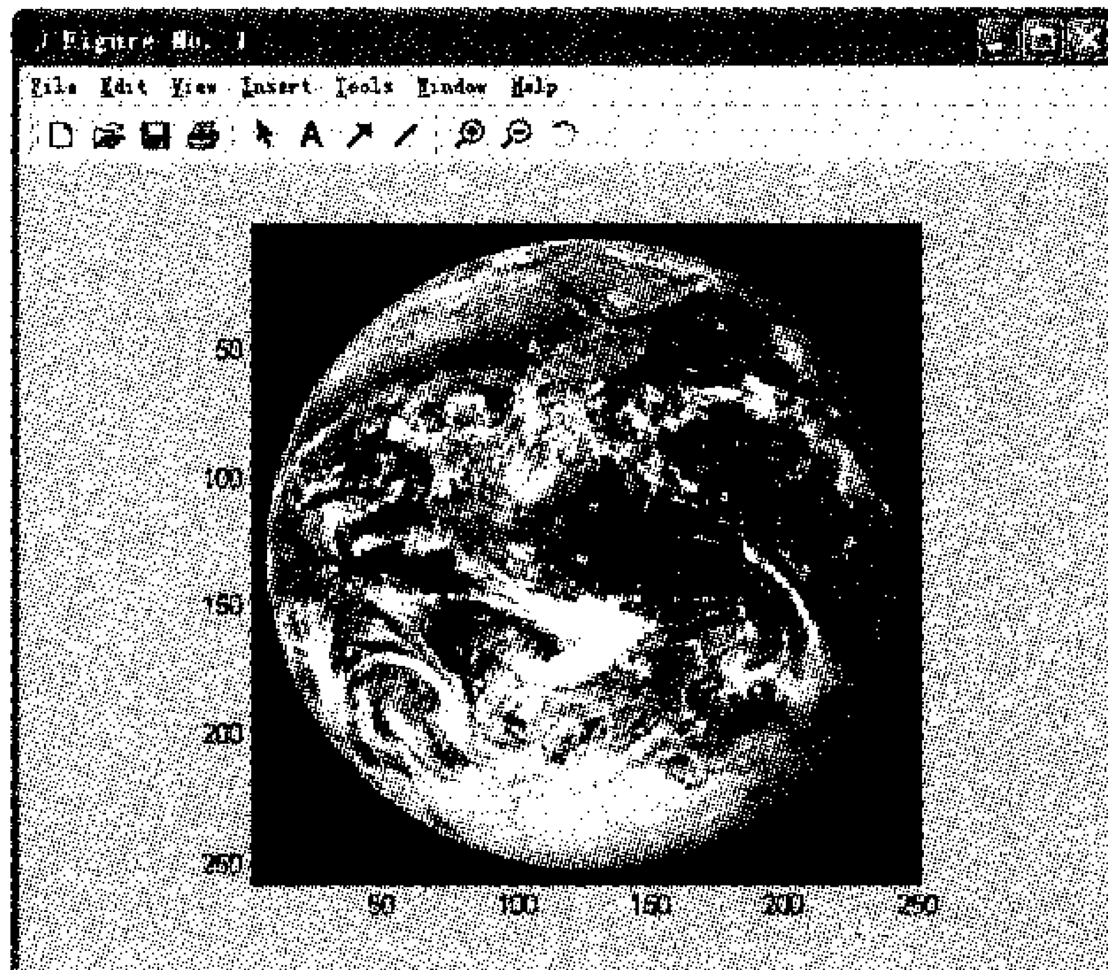


图 4-53　图像的显示

4．line 及其相关命令

一般不用低级命令 line 来直接创建线条对象，而用高级命令，如 plot。然而有时也用 line 命令来操作已有的线条和画新线条。和高级命令 plot 相比，line 命令只是画线条，而 plot 命令除了画线条外，还可以做一些其他的事情，如替换坐标轴等。

在 MATLAB 中用 line 函数创建线条对象，该函数调用格式如表 4-41 所示。

表 4-41　line 函数调用格式

调 用 格 式	说　明
line(X,Y)	在当前的坐标轴中画出由向量 *X* 和 *Y* 定义的线条
line(X,Y,Z)	在三维空间中画出由 *X*，*Y*，*Z* 定义的线条
line(X,Y,Z,'ProName','ProVal',...)	画出由参数 *X*，*Y*，*Z* 确定的线条，其中将指定属性 *ProName* 设置为 *ProVal*
h = line(...)	返回每一条线的线对象对应的句柄属性值向量

例程 4.50 为 line 函数应用示例。

例程 4.50　line 对象应用示例

```
t=0:pi/20:2*pi;
hline1=plot(t,sin(t),'k');
hline2=line(t+0.06,sin(t),'LineWidth',3,'Color',[.6 .6 .6])
```

例程 4.50 执行结果如图 4-54 所示。

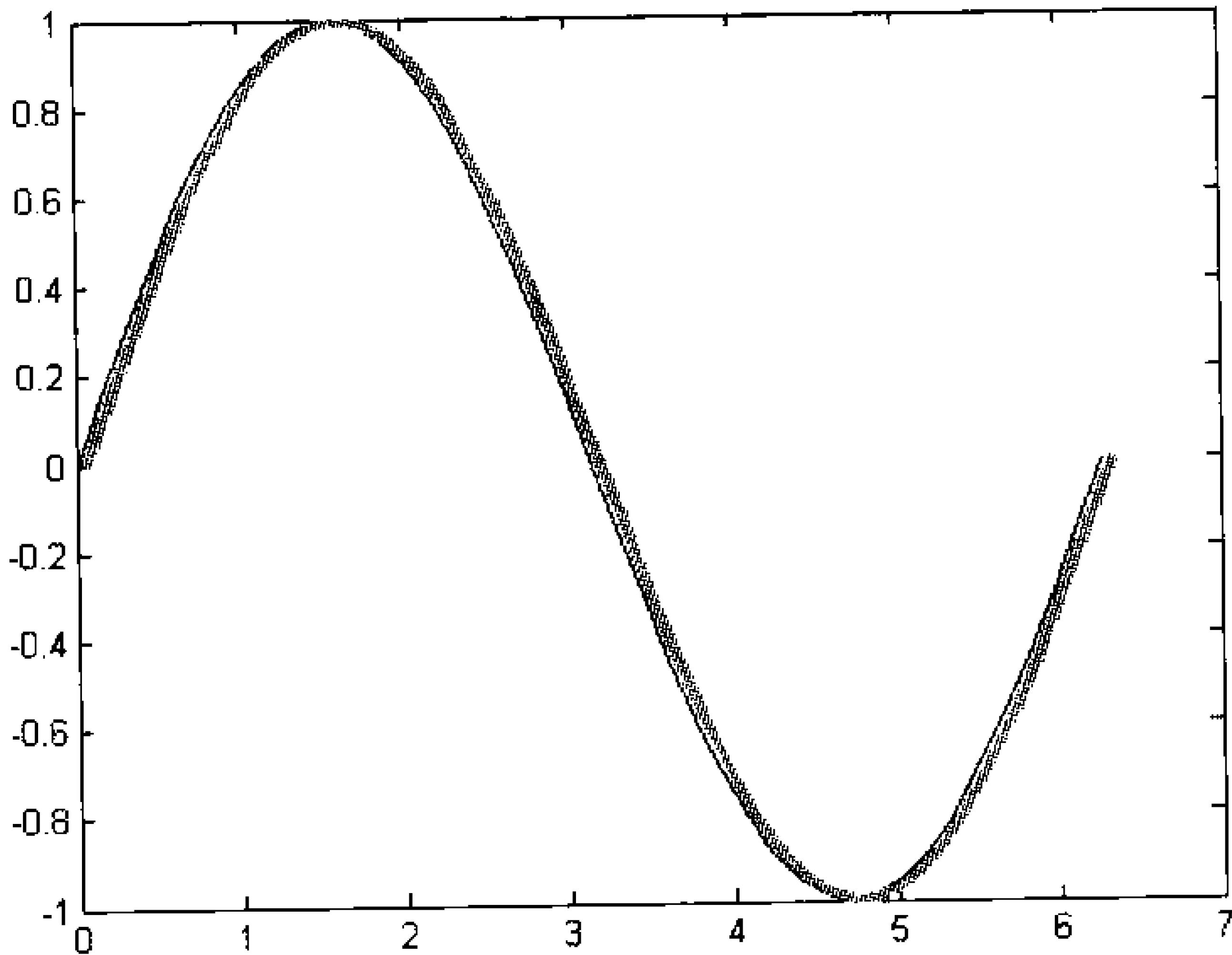


图 4-54　line 应用示例

5．text 及其相关命令

在 MATLAB 的图形对象中往往针对一定的需要，给出对象的注释。文本对象用 text 函数创建。text 函数是创建文本图形对象的低层函数，以文本的形式把字符串放置在指定的位置上。该函数的使用方法在前面已经介绍过。例程 4.51 为绘制图形后，利用 text 标出每一个转折点的位置坐标。

例程 4.51　文本对象示例

```
t=1:10;      %定义欲标示值的时间点位置
y=rand(size(t));      %产生一个与 t 大小相同的随机数 y
hold on
for k=1:length(t)
```

```
        text(t(k)+0.15,y(k),num2str(k));                 %依次标出 t 值
        text(t(k)+0.25,y(k),',');                        %依次标出 “,”
        text(t(k)+0.35,y(k),num2str(y(k),'%.2f'));       %依次标出 y 值
end
p=plot(t,y,t,y,'o');
%设置折线的封闭记号线及其边界颜色
set(p,'MarkerFaceColor','r','MakerEdgeColor','b');
hold of
```

执行结果如图 4-55 所示。

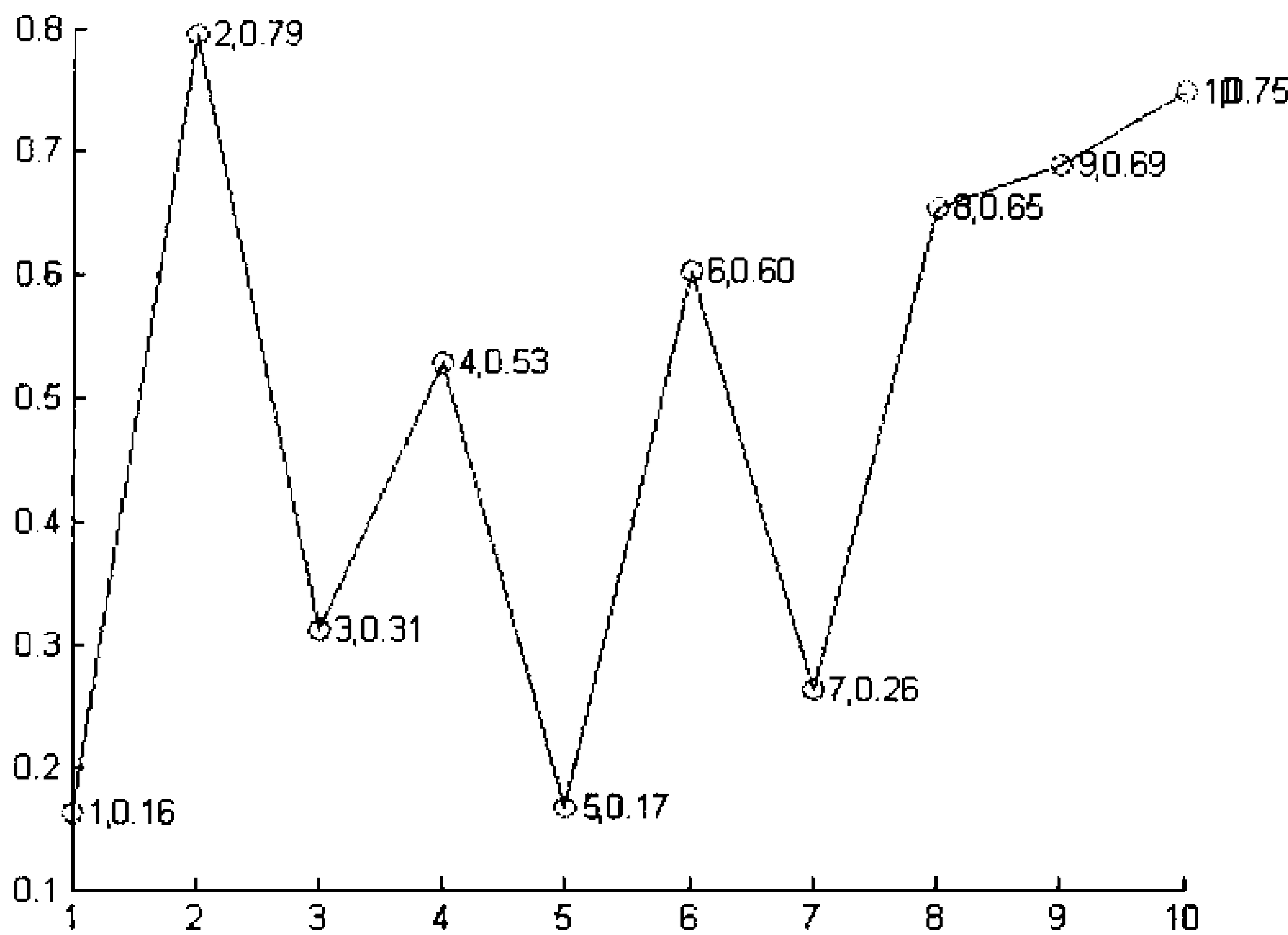


图 4-55 标出转折点的位置坐标

6．patch 及其相关命令

patch 由一个或多个多边形顶点坐标组成。用户可以指定 patch 的颜色与照明模式。patch 函数可以 patch 图形对象。注意，patch 函数不像其他高级区域建立函数（如 fill 或 area 函数），因此它没有检验 figure 窗口与 axes 的 NextPlot 属性，只是简单地将 patch 对象添加到当前坐标轴中而已。patch 函数调用格式如表 4-42 所示。

表 4-42 patch 函数调用格式

调用格式	说明
patch(X,Y,C)	在当前的坐标轴中添加二维填充的 patch 对象
patch(X,Y,Z,C)	产生三维的 patch 对象
patch(FV)	使用结构 *FV* 来建立 patch 对象
patch(...,'ProName','ProVal',...)	在二维或三维空间中，通过 patch 的属性名称与属性值来建立 patch 对象
patch('ProName','ProVal',...)	使用此种格式可以让用户忽略 patch 颜色的指定
h = patch(...)	返回使用 patch 函数所建立的 patch 对象句柄值

例程 4.52 为使用 patch 函数产生多边形。

例程 4.52 patch 函数示例

```
ver=[0 0 0;1 0 0;1 1 0;0 1 0];    %定义各顶点的坐标
face=[1,2,3,4];    %定义 Vertices 属性指定的坐标所连接的面的顺序
%因此顺序可以指定为红、黄、紫、绿
```

```
face_vercd=[1 0 0;0 1 0;1 0 1;1 1 0];
%对多边形的边沿颜色、表面颜色等属性进行设置
patch('vertices',ver,'faces',face,'facevertexcdata',face_vercd,...
        'facecolor','w','edgecolor','flat','marker','o',...
        'markerfacecolor','flat','linewidth',3);
```

得到的结果如图 4-56 所示。

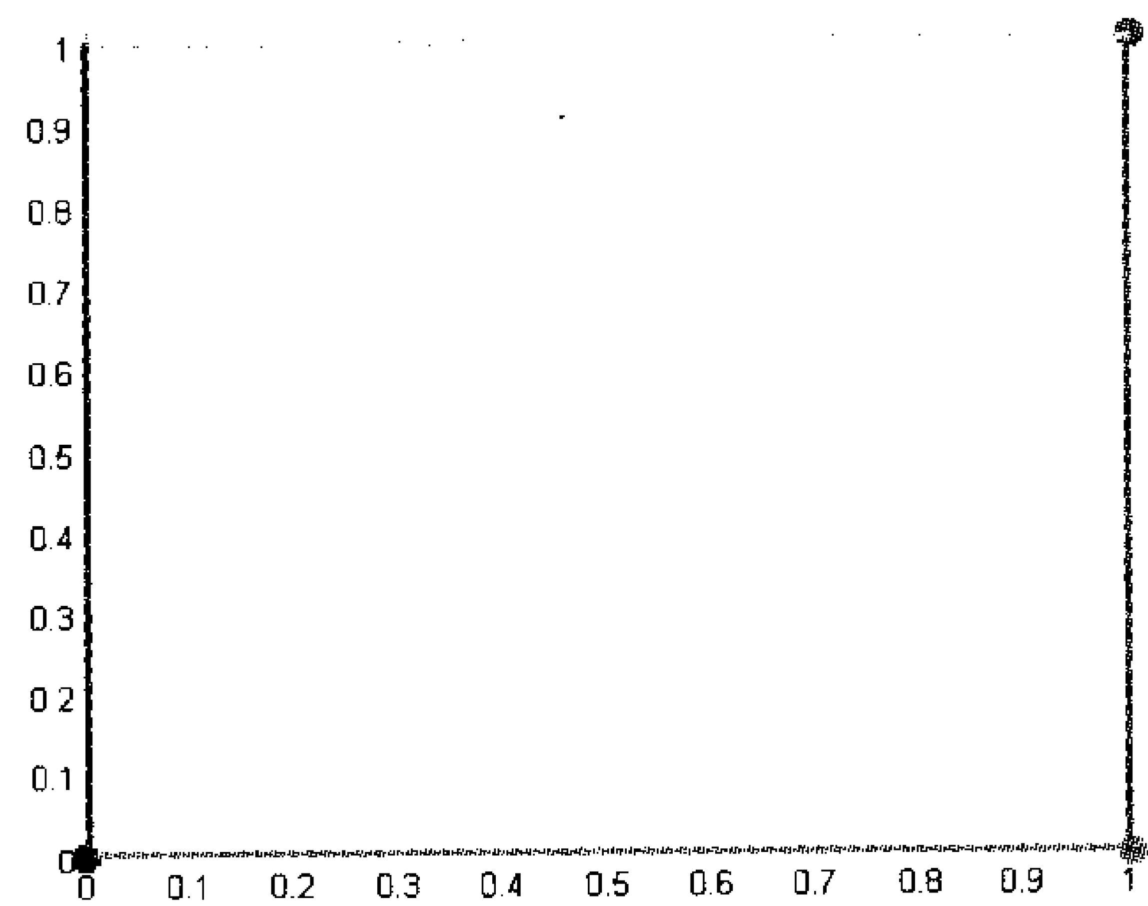


图 4-56　patch 函数产生多边形

7．surface 及其相关命令

Surface 对象是由矩阵数据所在的列索引值为 X 坐标，而行索引值为 Y 坐标，并且矩阵的每个元素值为 Z 坐标，这些指定的点所绘制的空间曲面。surface 函数可以建立一个 surface 对象。该函数调用格式如表 4-43 所示。

表 4-43　surface 函数调用格式

调 用 格 式	说　　明
surface(Z)	画出由矩阵 Z 所定义的曲面，其中 Z 是定义在一个几何矩形区域网格线的单值函数
surface(Z,C)	画出颜色由矩阵 C 指定且曲面由矩阵 Z 所指定的空间曲面
surface(X,Y,Z)	使用颜色 $C=Z$，因此该颜色能适当地反映曲面在 x-y 平面上的高度
surface(X,Y,Z,C)	曲面由参数 X、Y 与 Z 指定，颜色由 C 指定
patch('ProName','ProVal',...)	指定曲面的属性，对曲面进行微控制
h =surface(...)	返回建立 surface 对象的句柄值

例程 4.53 为 surface 应用示例。

例程 4.53　surface 函数示例

```
load clown      %载入图像数据
%创建 surface 对象并对其属性进行操作
surface(peaks,flipud(X),...
        'FaceColor','texturemap',...
        'EdgeColor','none',...
        'CDataMapping','direct')
colormap(map)
view(-35,45)
```

得到的结果如图 4-57 所示。

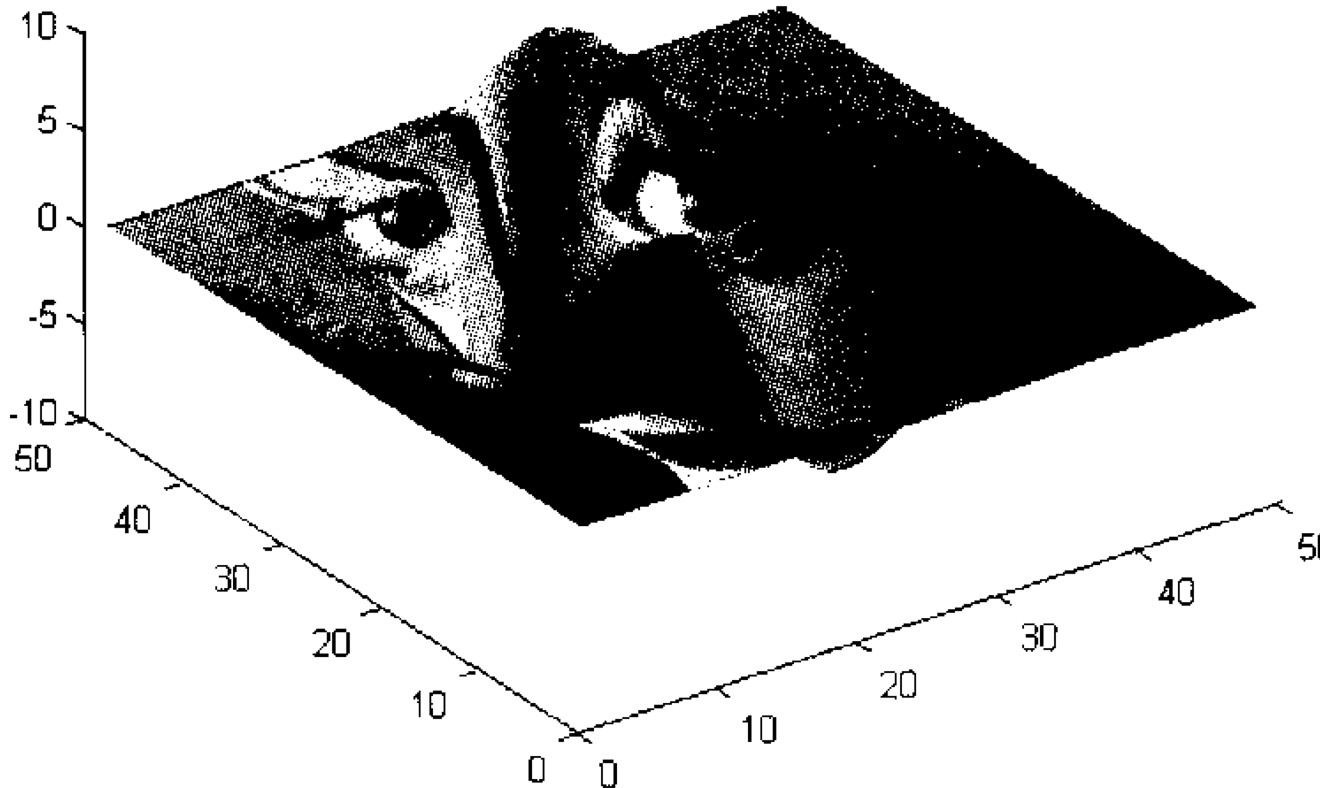


图 4-57 surface 函数产生空间曲面图

表 4-38 中所列的其他图形对象的创建及其属性，请读者参阅 MATLAB help 文档。

4.4.3 图形对象句柄的获取

要设置图形对象属性值，就要先知道该对象的句柄。因此，掌握对象句柄的获取方法很重要，获取的方法有以下几种。

1）从图形创建指令获取句柄

所有高层或低层指令（在此用 GraphicCommand 表示）都能通过以下格式产生句柄。

```
H_GC=GraphicCommand(……)        %绘图的同时给出句柄的调用指令 H_GC
```

2）追溯法获取句柄

若一个句柄对象已知，那么可以用如下格式追溯获得其“父”或“子”的句柄。

```
H_pa=get(H_known,'parent')         %获取 H_known 对象之“父”的句柄
H_ch=get(H_known,'Children')       %获取 H_known 对象之“子”的句柄
```

3）当前对象句柄的获取

MATLAB 有如下 3 个专门获取图像句柄的指令：前两个是直接指令式的；后一个必须与鼠标配合使用。

```
gcf                                %返回当前图形窗口的句柄
gca                                %返回当前轴的句柄
gcf                                %返回最近被鼠标单击的图形对象的句柄
```

4）根据对象特性获取句柄

利用对象特性搜索对象句柄可以得到较高的搜索速度。用到的 MATLAB 函数为 findobj。该函数调用格式如表 4-44 所示。

表 4-44 findobj 函数调用格式

调用格式	说明
H=findobj	返回根（Root）对象与其所有子对象的句柄值
H=findobj(h)	返回 *h* 变量的句柄值
H=findobj(' ProName',ProVal)	依据对象的属性名称与属性值找出相匹配的对象句柄值。
H=findobj(ObHanles' ProName',ProVal)	依据限定的对象列表 *ObHandles*（如使用 gca）找出与对象的属性名称和属性值相匹配的对象句柄值
H=findobj(ObHanles' ProName',ProVal,…)	同上，但不搜索它们的子对象

用户可以通过“tag”属性，给对象一个“诨名”。此后，就可以通过“诨名”获取该对象的句柄。设置“诨名”有以下两种方法。

（1）创建时赋名，如下：

```
subplot(4，4，2)，plot(x,y,'tag','A');
```

（2）以 set 赋名，如下：

```
subplot(4,4,2)，h=plot(x,y)，set(h,'tag','A')。
```

如果屏幕上有多个图形窗口，且有的窗口又有多个子图，那么获得带“诨名”对象句柄的简单命令是 hax=findobj(0,'tag','A')，函数 findobj 返回符合所选判据的对象的句柄。它检查所有的 Children，包括坐标轴的标题和标志。如果没有对象满足指定的判据，findobj 返回空矩阵。

如例程 4.54，通过 findobj 查找对象的句柄值，将绘制的第一条曲线宽度设置为 5。

例程 4.54　findobj 函数示例

```
clf reset    %将窗口关闭，并重新设置其属性为默认值
t=linspace(0,2*pi);
plot(t,sin(t),t,cos(t));    %绘制图形，MATLAB 默认第一条线段为蓝色
b_handles=findobj(gca,'Color','b');    %返回当前坐标轴蓝色线段的句柄值
set(b_handles,'linew',5);
set(gcf,'position',[250 250 300 200],'name','findobj 范例');
```

执行结果如图 4-58 所示。

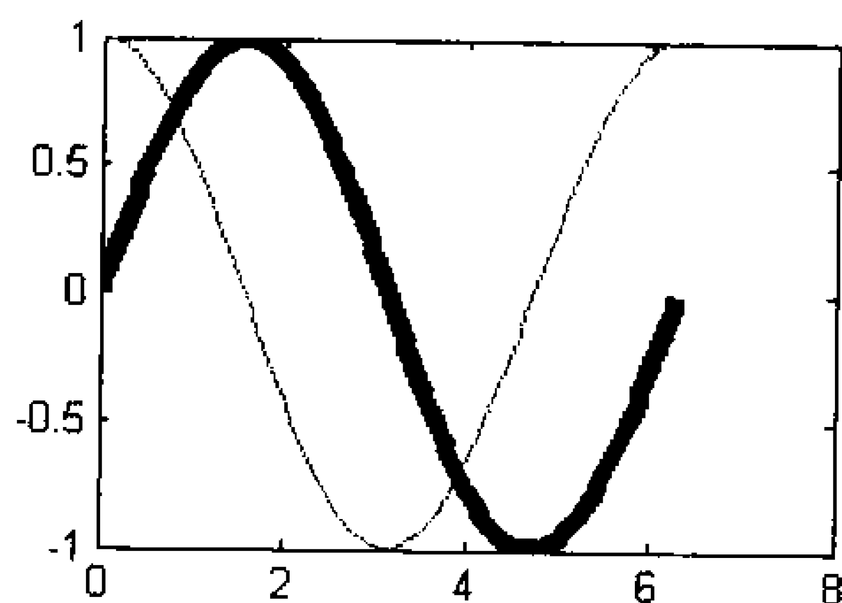


图 4-58　findobj 查找对象的句柄值

当然，findobj 也可以搭配 gcf、gca 和 gco 以便查找更为迅速。

获取图形句柄后，就可以对图形对象进行各种操作。其中包括删除图形对象，其函数为 delete。例如，delete(gca)将删除当前轴和它的所有子对象。

用 MATLAB 的低级图形命令编程时，经常要用句柄来查询对象的属性值，在这方面 MATLAB 提供了很多查询句柄的函数。但在编写 M 文件时，使用这些函数往往不是最好的方法。在创建对象时，用一个变量来保存句柄值，往往能提高编程和执行效率。

4.4.4　对象属性的获取

所有的图形对象都有属性，正是通过设置这些属性来定义或修改图形的特征。每个不同的对象都有和它相关的属性，用户可以获取并且改变这些属性而不影响同类型或不同类型的其他对象。

在 MATLAB 中通过 get 函数可以取得图形对象的属性值。

get 函数主要有如下几种调用格式，如表 4-45 所示。

表 4-45　get 函数调用格式

调用格式	说　明
ProVal=get(H,'ProName')	获得句柄为 *H* 的对象中名为“*ProName*”的属性的值
get(H)	显示句柄为 *H* 的对象的所有属性名及当前的取值
ProVal=get(H)	返回一个结构，结构的每个域名就是句柄为 *H* 的对象的所有属性名，每个域又包括属性的值
ProVal=get(H,'Factory<ObjectType><ProName>')	返回其所有可以由用户设置默认值的属性的“出厂值”，所谓“出厂值”指未经过任何用户改动的最初的默认值
ProName=get(0,'Default<ObjectType><ProName>')	返回默认的属性值

试用 mesh 函数绘制 peaks 函数并返回句柄值 m，再由 get 对该句柄值做一些简单的属性操作，如例程 4.55。

例程 4.55　get 函数示例

```
>> m=mesh(peaks(30));    %返回网目图的句柄值 m
>> get(m,'MarkerSize')    %获得 m 对象的 MarkerSize 属性的值
ans =
      6
>> ph=get(m,'Parent');    %返回网目图的父对象，即坐标轴
>> get(ph,'ZTickLabel')    %获得坐标轴的 Z 轴卷展栏
ans =
-10
-5
0
5
10
```

表示返回绘制 peaks 网目图记号的大小为 6，并且父对象 Axes 的 *Z* 坐标刻度卷展栏为 −10、−5、0、5、10。

4.4.5　对象属性的设置

MATLAB 中通过 set 函数用来设置对象的属性值，它的调用格式如表 4-46 所示。

表 4-46　set 函数调用格式

调用格式	说　明
set(H,' ProName',ProVal)	把句柄为 *H* 的对象中名为“ProName”的属性的值设置为“*ProVal*”
set(H,a)	把属性值赋给和域名相同的属性
set(H,PN,PV)	把句柄中指定的所有对象的属性设置为“PV”中的指定值
set(H,'ProName1',ProValue1,…)	同时设置多个属性值
A=set(H, 'ProName')、 set(H, 'ProName')	返回或显示句柄为 *H* 的对象的指定属性的值
A=set(H)、 set(H)	返回或显示句柄为 *H* 的对象的所有属性和可能的取值
set(h,'DefaultObjectTypeProName',ProVal)	设置对象属性的默认值

先通过鼠标单击“设置字符串”这段文字放置的位置，指定位置后，就会立即将该字符串旋转一定度数，并将原来显示的“GUI 测试”字符串改为“成功”，最后再更改坐标轴的刻度显示如例程 4.56。

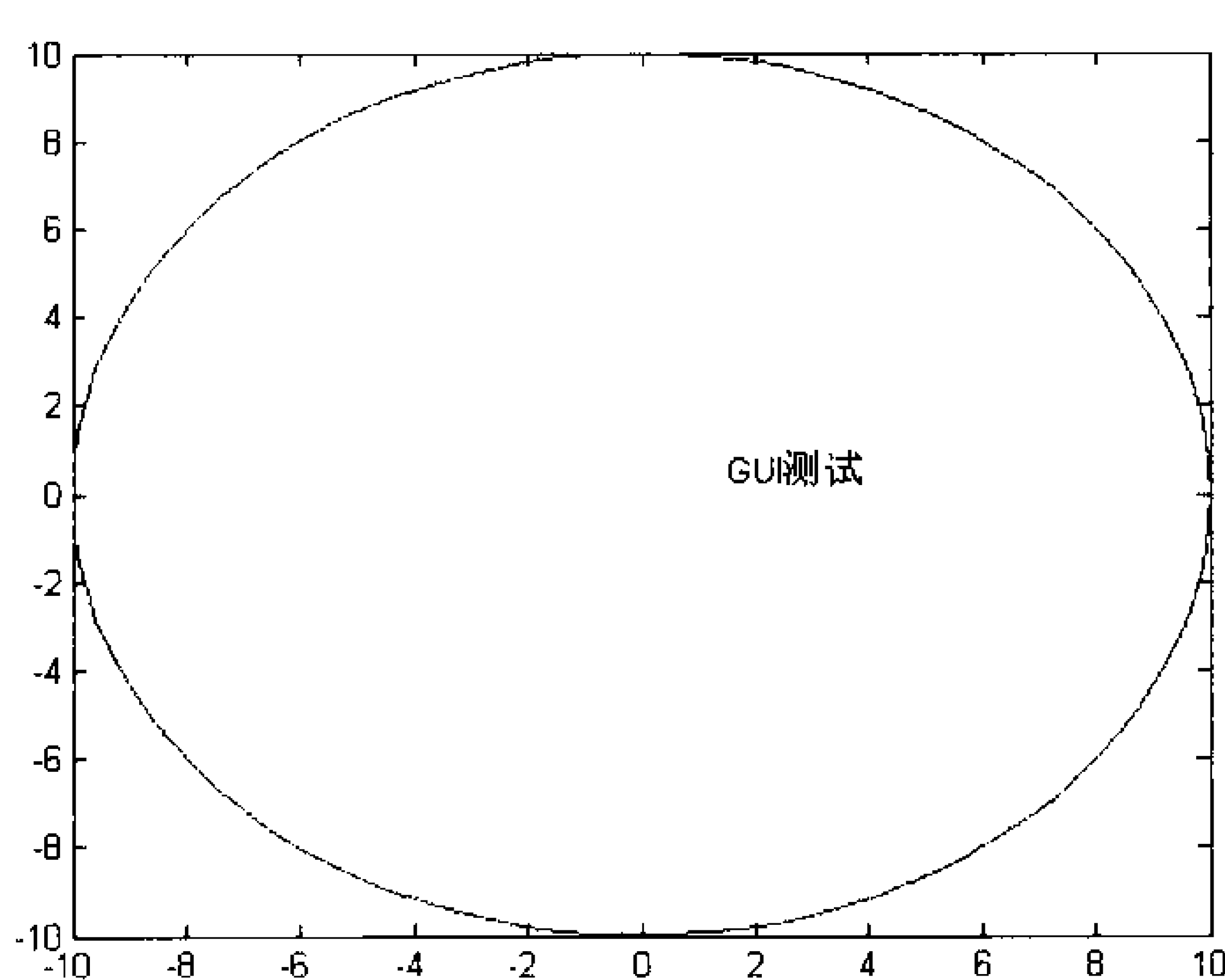
例程 4.56　set 函数示例

```
x=linspace(0,2*pi);
x1=10*cos(x);
y=10*sin(x);
plot(x1,y);
t=text(1.5,0.5,'GUI 测试');        %建立一个句柄值为 t 的 text 对象并显示“GUI 测试”

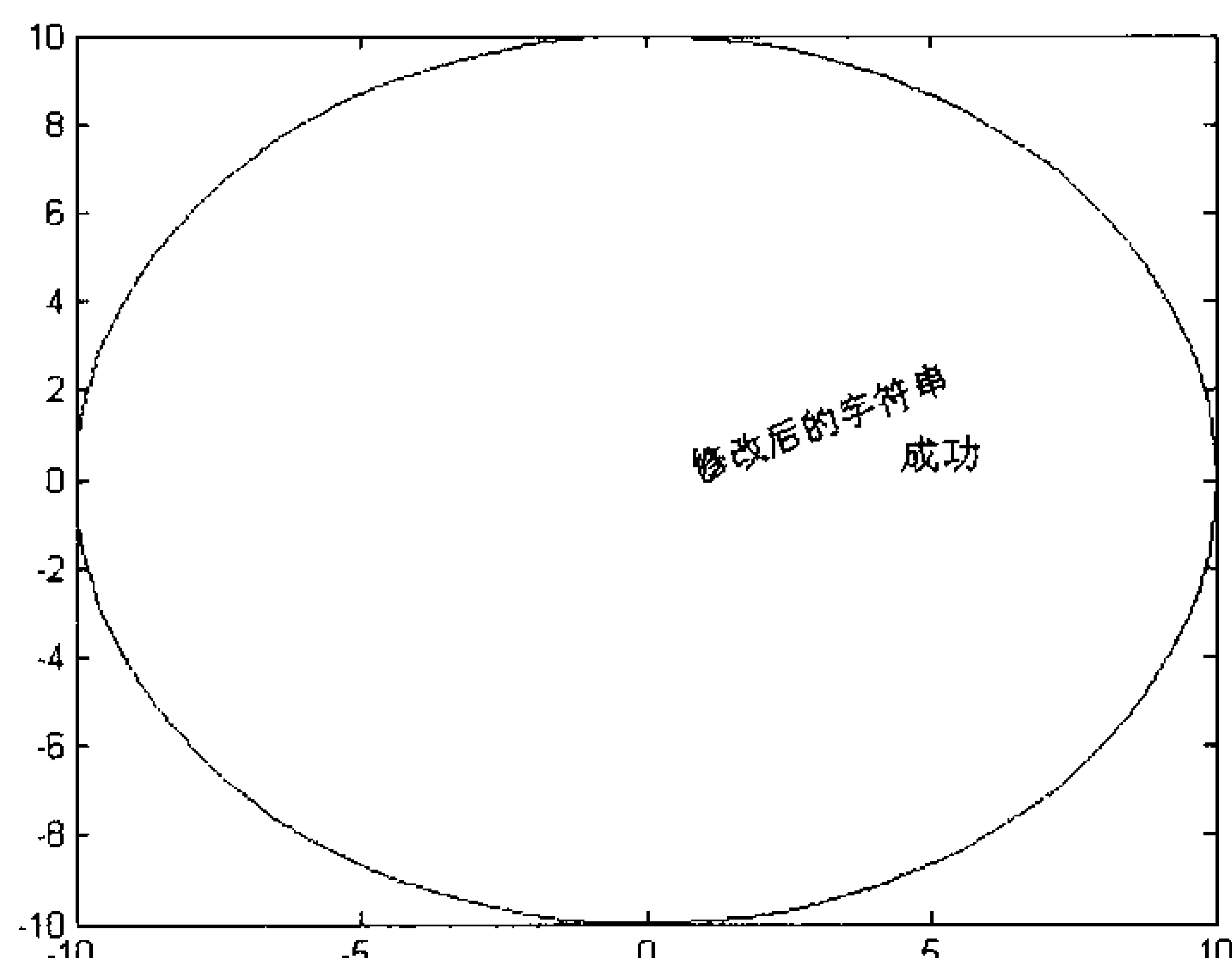
gt=gtext('设置字符串');    %建立一个句柄值为 gt 的 gtext 对象并显示“设置字符串”
%将句柄值为 gt 的 gtext 对象旋转 20 度，并将字符串内容改为“修改后的字符串”
set(gt,'String','修改后的字符串','Rotation',20);

%重新指定句柄值为 t 的 text 对象位置，并将字符串内容改为“成功”
set(t,'Position',[4.5 0.5],'String','成功')
set(gca,'XTick',[-10:05:10])    %设置显示的刻度范围
set(gea,'YMinorTick','on')    %打开小刻度显示
```

得到的结果如图 4-59 所示。



（a）选取文字放置位置



（b）完整显示结果

图　4-59

若并不了解需要输入哪些属性值时，则可将 set 与 get 当做辅助工具。如使用 set 仅输入该对象的属性名称，即 set(H,' ProName')，则 MATLAB 就会立即返回句柄值为 H 所对应的对象的属性名称下所有可设置的属性值。假设要查询绘图窗口对象 Units 属性的所有可设置的属性值，则可输入：

```
>> set(gcf,'Units')
[ inches | centimeters | normalized | points | {pixels} | characters ]
```

此时 MATLAB 会告知用户应该输入哪些 Units 属性值，其中大括号{}所括住的属性值就是当前的默认值，因此当前绘图窗口的单位为像素（Pixel）。对于需要输入一个特定数值的属性名称而言，使用 set 是无法查询可设置的属性值的，因此，当没有指定一个特定数值时，MATLAB 就会显示一个信息要求输入值。

```
>> set(gcf,'Position')
```

```
A figure's "Position" property does not have a fixed set of property values.
```

以上表示 Position 属性为设置大小与位置，因此并无固定的属性值可以设置。

为了形象地说明 get 与 set 函数的用法，再看看下面的例子。

例程 4.57 句柄对象属性查看示例

```
Hf_fig =        1
>> Hl_line=line   %画线
Hl_line =
  191.0011
>> set(Hl_line);   %列出其属性名称和属性值
      Color
      EraseMode: [{normal}| background|xor|none]
      LineStyle: [{-}|--|:|-.|none]
      LineWidth
      Marker: [+|o|*|.|x|square|diamond|v|^|>|<|pentagram|hexagram|{none}]
      MarkerSize
      MarkerEdgeColor: [none|{auto}]-or-a ColorSpec.
      MarkerFaceColor: [{none}|auto]-or-a ColorSpec.
      XData
      YData
      ZData

      ButtonDownFcn: string -or- function handle -or- cell array
      Children
      Clipping: [{on}|off]
      CreateFcn: string -or- function handle -or- cell array
      DeleteFcn: string -or- function handle -or- cell array
      BusyAction: [{queue}|cancel]
      HandleVisibility: [{on}|callback|off]
      HitTest: [{on}|off]
      Interruptible: [{on}|off]
      Parent
      Selected: [on|off]
      SelectionHighlight: [{on}|off]
      Tag
      UIContextMenu
      UserData
      Visible: [{on}|off]

>> get(Hl_line);   %列出属性名称和当前属性值
      Color = [0 0 0]
      EraseMode = normal
      LineStyle = -
      LineWidth = [0.5]
      Marker = none
      MarkerSize = [6]
      MarkerEdgeColor = auto
      MarkerFaceColor = none
      XData = [0 1]
```

```
YData = [0 1]
ZData = []

BeingDeleted = off
ButtonDownFcn =
Children = []
Clipping = on
CreateFcn =
DeleteFcn =
BusyAction = queue
HandleVisibility = on
HitTest = on
Interruptible = on
Parent = [161.001]
Selected = off
SelectionHighlight = on
Tag =
Type = line
UIContextMenu = []
UserData = []
Visible = on
```

在本例中，所创建的线条中的“Parent”属性就是包含线条的坐标轴的句柄，而且所显示的图形列表被分为两组。在空行上的第一组，列出了该对象的独有属性，而空行下的第二组列出所有的对象共有的属性。函数 set 和函数 get 返回不同的属性列表。函数 set 只列出可以用 set 命令改变的属性，而 get 命令列出所有对象的属性。在上面的例子中，函数 get 列出了“Children”和“Type”属性，而 set 命令却没有。这一类属性只读，但不能被改变，它们叫做只读属性。

与每一个对象有关的属性数目是固定的，但不同的对象类型有不同数目的属性。像上面所显示的一样，一个线条对象列出了 16 个属性，而一个坐标轴对象列出了 64 个属性。显然，透彻地说明和描述所有对象类型的全部属性超出本书的范围。但是，以后本书要详细讨论其中的很多属性，并且列出全部属性。

例程 4.58　图像句柄属性使用示例

```
%先创建橘黄色画线，线的颜色[1,0.5,0]
>> x=-2*pi:pi/40:2*pi;
>> y=sin(x);
>> Hl_sin=plot(x,y)
Hl_sin =
   159.0024
>> set(Hl_sin,'Color',[1 .5 0],'LineWidth',3)
%现在加一个浅蓝色的 cosine 曲线
>> z=cos(x);
>> hold on
>> Hl_cos=plot(x,z);
>> set(Hl_cos, 'Color',[.75   .75   1])
>> hold off
%加上一个标题并且使字体比正常大一些
```

```
>> title( 'Handle Graphics Example' )
>>   Ht_text=get(gca,'Title' )
Ht_text =
   161.0020
>> set(Ht_text,'FontSize',16)
```

输出结果如图 4-60 所示。

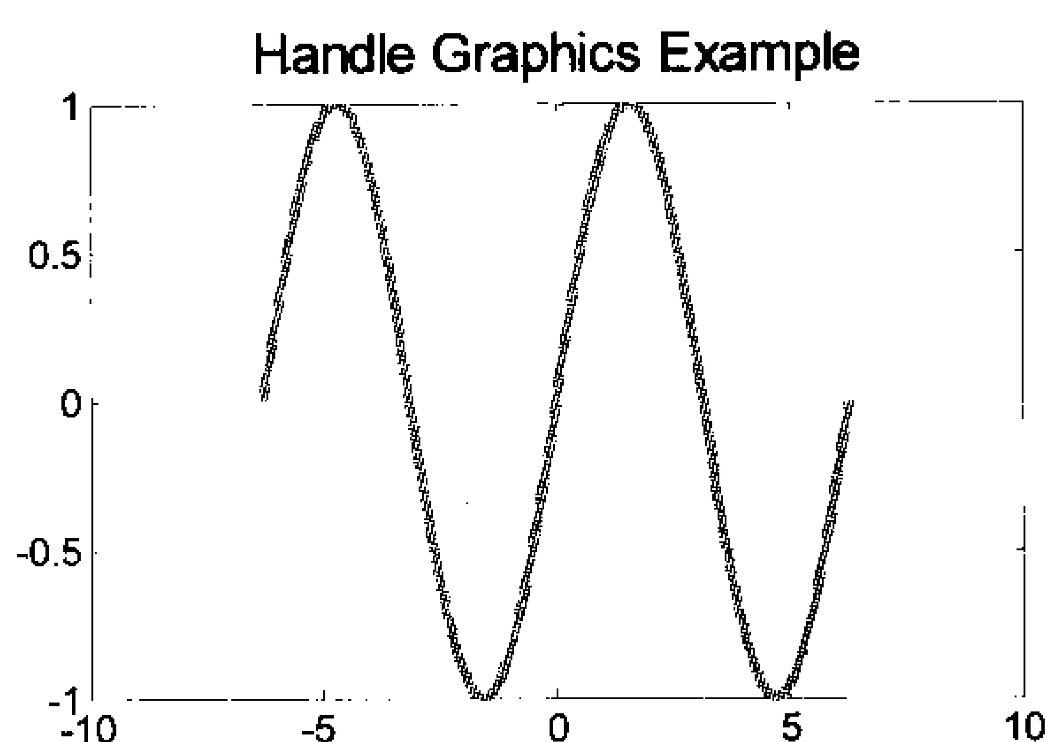


图 4-60　改变图形句柄属性示例

每一个对象都含有“Parent”属性和“Children”属性，该属性包含属于派生对象的句柄。画在一组坐标轴上的线，具有“Parent”属性值的坐标轴对象的句柄，而“Children”属性值是一个空矩阵。同时，这个坐标轴对象具有当做“Parent”属性值的图形句柄，而“Children”属性值是线条对象的句柄。标题字符串和坐标轴的标志不包含在坐标轴的“Children”属性值里，而是保存在“Title”、“Xlabel”、“Ylabel”和“Zlabel”的属性内。创建坐标轴对象时，这些文本对象就建立。title 命令设置当前坐标轴内标题文本对象的 String 属性。最后，标准 MATLAB 的函数 title、xlabel、ylabel 和 zlabel 不返回句柄，而只接受属性和数值参量。例如，下面的命令给当前图加一个 24 点的绿色标题。

```
>> title(' This   is   a   title. ' , 'Fontsize' ,24, 'Color' , ' green')
```

另外，MATLAB 还提供了 reset 函数用来重新设置图形对象的属性为它们的默认值。该函数的使用方法为：

```
reset(h)
```

但是，如果 *h* 为一个 figure 对象，则 reset 函数无法重新设置 Position、Units、PaperPosition 与 PaperUnits 等属性；如果 *h* 为一个 axes 对象，则 reset 函数无法重新设置 Position 与 Units 属性。如下语句为重新设置当前坐标轴的属性。

```
>>reset(gca)
```

4.5　MATLAB 图像显示技术

图像显示是一种特殊的图形绘制。MATLAB 提供了一系列命令和函数用于显示和处理图像。在 MATLAB 中，图像数据通常被创建或保存为标准的双精度浮点数，有时也可以创建或者保存为 8 位或 16 位的无符号整数。MATLAB 能够读写多种格式的图像文件，也可以用 load 和 save 命令来将图像数据保存在 MAT 文件中。

本章主要讨论了 MATLAB 图像类型、图像读取与显示。因为本节非本书重点，所以没有涉及更详细的信息，如果读者需要了解更多 MATLAB 图像方面的知识，请参阅 MATLAB

帮助文档或其他资料。

4.5.1 图像简介

在 MATLAB 中，图像（Image）通常由数据矩阵和色彩矩阵组成。根据图像着色方法的不同，MATLAB 图像可以分为 3 类：索引图像（Indexed Image）、亮度图像（Intensity Image）和真彩色图像（True Color or RGB Image）。

索引图像是带有颜色表矩阵的，图像数据矩阵中的数据通常被解释成指向颜色表的矩阵的索引号。图像颜色表矩阵可以是上一节讲到的任何有效的颜色表：即任何包含了有效 RGB 数据的 $m\times3$ 的数组。如果索引图像的图像数据数组为 $X(i,j)$，颜色表数组为 cmap，则每个图像像素 $P_{i,j}$ 的颜色就是 cmap(X(i,j),:)。这要求 X 中的数据值必须是位于[1 length(cmap)]范围之内的整数。如果用户已经获得图像数据和颜色表，可以使用下面的命令显示这幅图像：

```
>>image(X);colormap(cmap)
```

亮度图像的图像数据矩阵通常表示该图像的亮度值。该类型图像通常用于显示由灰度或单色颜色表染色的图像，有时也用于其他颜色表染色的图像。亮度图像对数据范围没有要求，不一定要像索引图像那样位于[1 length(cmap)]范围之内。用户可以指定亮度图像的数据范围，并且将其作为指向颜色表的索引。如下面的例子：

```
>>imagesc(X,[0 1]);colormap(gray)
```

将 X 的值限制在[0 1]之间，并将 0 指向颜色表的第一个颜色，将 1 指向颜色表的最后一个颜色，介于 0 和 1 之间的数据被用来作为指向颜色表中其他颜色的索引。如果在上面的语句中省略[0 1]，则意味着不对 X 进行限定，也就是说，X 的数据范围是[min(min(x)) max(max(X))]。

真彩色（也叫 RGB）图像通常由一个包含有效 RGB 值的 $m\times n\times3$ 的数组创建。该数组的行和列声明了像素的位置，也声明了图像中每一个像素的颜色值。也就是说，像素 $P_{i,j}$ 将用 X(i,j,:)所声明的颜色绘制。由于真彩色图像已经将颜色信息包含在图像数据中，因此它不需要颜色表。如果计算机硬件不支持真彩色图像（例如，它只有一块 8 位显卡），那么 MATLAB 就利用颜色近似和抖动来显示图像。真彩色图像的显示比较简单，如下例所示：

```
>>image(X)
```

其中，X 是一个 $m\times n\times3$ 的真彩色图像。X 可以包含双精度数据，也可以包含 unit8、unit16 类型的数据。

如果事先不知道图像的类别，那么就先用 imfinfo 指令获取该图像的信息，然后再进行读操作。图像着色类型不同，其显示和写入指令也不同。以下命令用来获取图像文件的特征数据（特别是着色类型 ColorType）。

```
>>infinfo(FileName)
```

指令 imfinfo 将产生一个构架数组。不管数组的大小如何，在构架上都有一个名为 ColorType 的域，域中存放着如下 3 种“图像着色类型字符串”。

- indexed：变址着色的图像。
- grayscale：灰度着色的图像。
- truecolor：真彩着色的图像。

Parameter/Value 用来修改对象属性。常用的 Parameter/Value 随图像格式不同而不同，具体情况如表 4-47 所示。

表 4-47　常用的 Parameter/Value

格　式	Parameter	Value	默 认 值
JPEG	Quality	[0,100]之间的任何数	75
TIFF	Compression	“none”，“packbits” 对二维图可选 “ccitt”	二维图像用 “ccitt”；其余用 “packbits”
	Description	任何字符串	空串
HDF	Compression	“none”，“rle”，“jpeg”	“rle”
	WriteMode	‘overwrite’,’append’	“overwrite”
	Quality	[0,100]之间的任何数	75

4.5.2　图像的读取

不同的类型图像有自己固定的数据格式。要在 MATLAB 下使用其他软件中使用的图像，需要用 imread 函数读取该图像。这实际上也是一个数据转换的过程，即把该图像的数据转换成 MATLAB 图像的数据格式。函数 imread 的调用格式如表 4-48 所示。

表 4-48　imread 函数调用格式

调 用 格 式	说　明
A=imread(filename,fmt)	返回存放图像的变量名 A
[*X*,MAP]=imread(filename,fmt)	返回图像的数值存放矩阵 X 和颜色矩阵 MAP
[...]=imread(filename)	返回图像的信息

其参数含义如表 4-49 所示。

表 4-49　函数 imread 中参数的含义

参 数 名 称	描　述
filename	图像的文件名
Fmt	指定图像的类型，可以为：JPEG（jpg 或 jpeg）、TIFF（tif 或者 tiff）、BMP(bmp)、PNG（png）、HDF（hdf）、PCX（pcx）和 XWD（xwd）
A	由图像文件中读出并转化成 MATLAB 可识别的图像格式的数据
X	保存索引图像数据的数组
MAP	保存相关颜色映像的数组

下面为 imread 函数应用示例。

例程 4.59　imread 函数示例

```
>> A=imread('bear.jpg','jpg');
>> size(A)
ans =
    174    193     3
```

```
>> A=imread('bear','jpg');
>> size(A)
ans =
   174    193     3
>> A=imread('bear.jpg');
>> size(A)
ans =
   174    193     3
```

可以通过上述的 3 种方式来读取真彩色文件“bear.jpg”。读者可以看到三维数组“*A*”有 3 个面，它们依次为 R、G、B 3 个颜色，而面上的数据则分别是这 3 种颜色的强度值，面中的元素对应于图像中的像素点，因而面中的行数和列数与图中像素的行数和列数是一致的。

MATLAB 中的函数 imwrite 用于把图像输出到文件，调用格式如表 4-50 所示。

表 4-50　imwrite 函数调用格式

调 用 格 式	说　明
imwrite(A,filename,fmt)	将变量 *A* 以 *fmt* 格式存为 *filename*
imwrite(...,filename)	将当前图像矩阵以文件名 *filename* 存储
imwrite(...,'ProName','ProVal',...)	根据属性名'*ProName*'的值另存图像数据

其中参数 filename 和 fmt 与函数 imread 相同。而在第三条命令中，参数随着 fmt 的改变而改变，具体请参阅 MATLAB 帮助文件。

MATLAB 支持的一些常用图像/图形格式如表 4-51 所示。

表 4-51　MATLAB 支持的主要图像格式

格 式 名 称	描　述	可识别扩展名
TIFF	加标识的图像文件格式	.tif、.tiff
JPEG	联合图像专家组	.jpg、.jpeg
GIF	图形交换格式	.gif
BMP	Windows 位图	.bmp
PNG	可移植网络图形	.png
XWD	X Window 转储	.xwd
HDF	面向对象的自描述图像格式	.hdf
ICO	图标资源文件	.ico
CUR	光标资源文件	.cur
RAS	光栅图像位图	.ras
PCX	Window 画刷图形	.pcx
PGM	简便灰度图像	.pgm
PBM	简便位图格式	.pbm
PPM	简便像素图形	.ppm

对于上表中的 GIF 格式图像，imread 函数支持，但是 imwrite 函数不支持。

4.5.3 图像的显示

众所周知，数字图像是指将一幅二维的图像表示成一个数值矩阵，矩阵的元素被解释为像素的颜色值（或灰度值），或被解释为调色板颜色的索引号。为了显示由矩阵表示的数字图像，MATLAB 最一般的做法是将矩阵的每个元素对应到当前色谱的某个行标号，并取出该行的颜色值作为图像相应点的颜色。一般说来，每幅图的色调不同，因此作为图像必须有自己特殊的色图，这样才能真实地显示图像。

MATLAB 用函数 image 显示图像，该函数调用格式如表 4-52 所示。

表 4-52 image 函数调用格式

调 用 格 式	说 明
image(C)	把矩阵 C 作为一个图像画出
image(x,y,C)	在(x,y)确定的位置上画 C 的元素
image(x,y,C,'ProName','ProValue',...)	指定属性名和属性值，在(x,y)确定的位置上画 C 的元素
image('ProName','ProValue',...)	只接受属性名和属性值的输入
h = image(...)	返回刚生成的图片对象的句柄属性值向量

另一个与 image 函数类似的函数是 imagesc，它的命令格式与 image 一样。image(X)是将数据矩阵 X 的值直接作为索引号在色谱矩阵中提取 RGB 颜色值进行着色的。事实上，对于任何矩阵 X，image(X)可以生成一幅图像。如果 X 的元素的数值大小十分接近，或超出色谱矩阵的长度，那么 image(X)就不能有效地用图像表达矩阵 X，而函数 imagesc 就可以做到这一点。imagesc 在功能与 image 是一样的，只是按线性变换的方式计算索引号，即与 pcolor 使用的方法相同。于是，imagesc(X)生成的图像将受到 caxis 函数的影响。

4.6 动画制作

以动画来显示结果，除了可以让绘图更为生动之外，还可以立即比较出与原始图形的差异，深入强调绘图的重点所在，因此本节将介绍在 MATLAB 中绘制动画的几种方式与应用。

MATLAB 有很多建立动画的方式，有使用 comet 所产生的质点运动轨迹动画：通过 spinmap 设置系统色彩变化所产生的动画效果——以电影播放的方式产生动画效果；以及通过 drawnow 的对象动画方式，它们各有自己的优缺点，下面依次介绍这些方法。

4.6.1 以质点运动轨迹的方式呈现动画

质点运动轨迹动画（使用 comet、comet3）方式是最简单的动画产生方式，顾名思义，就是产生一个顺着曲线轨迹运动的质点来操作。下面介绍 comet 与 comet3 的使用方式。

表 4-53 函数 comet 和 comet3 调用格式

调用格式	说明
comet(y)	显示质点绕着向量 y 的动画轨迹（二维）
comte(x,y)	显示质点绕着向量 y 与 x 的动画轨迹（二维）
comet(x,y,p)	同上面的效果，但额外地定义轨迹尾巴线的长度 $p\times length(y)$，其中 p 是介于 0 和 1 之间的数，默认为 0.1

下面介绍这两个函数的应用示例。

例程 4.60 使用函数 comet 建立一个质点绕着圆轨迹运动的动画

```
>> t=linspace(0,2*pi,1000);
>> x=cos(t);
>> y=sin(t);
>> plot(x,y),axis square,hold on;      %画出图形，以便比较 comet 是否跟着轨迹走
>> comet(x,y,0.03)
```

得到的结果如图 4-61 所示。

comet3 与 comet 的做法类似，但 comet3 是用于三维的质点运动轨迹，因此必须加上 Z 轴资料。例程 4.61 为建立三维的曲线图形，并让质点沿着产生的轨迹运动。

例程 4.61 使用函数 comet3 建立一个质点绕着圆轨迹运动的动画

```
>> t=0:pi/50:10*pi;
>> plot3(sin(t),cos(t),t),axis square
>> comet3(sin(t),cos(t),t,0.5)
```

得到的结果如图 4-62 所示。

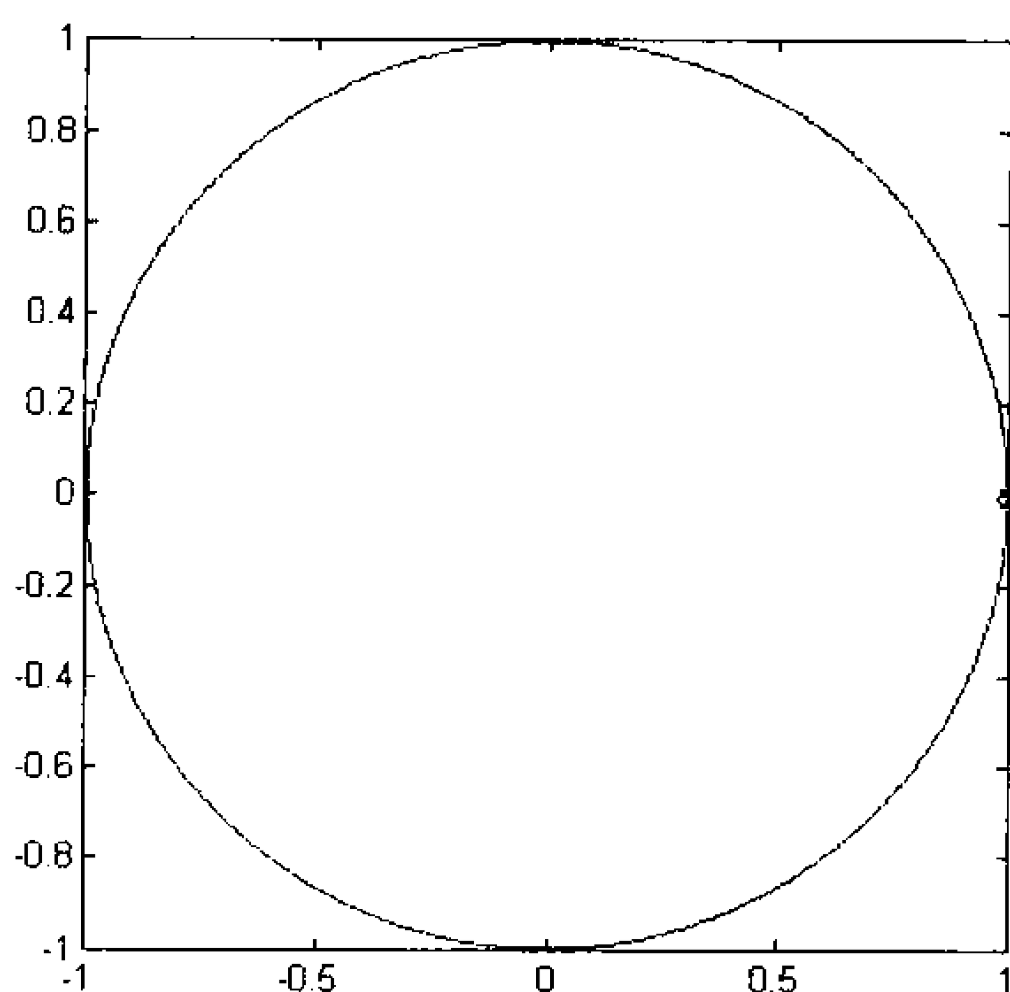


图 4-61 comet 绘图结果

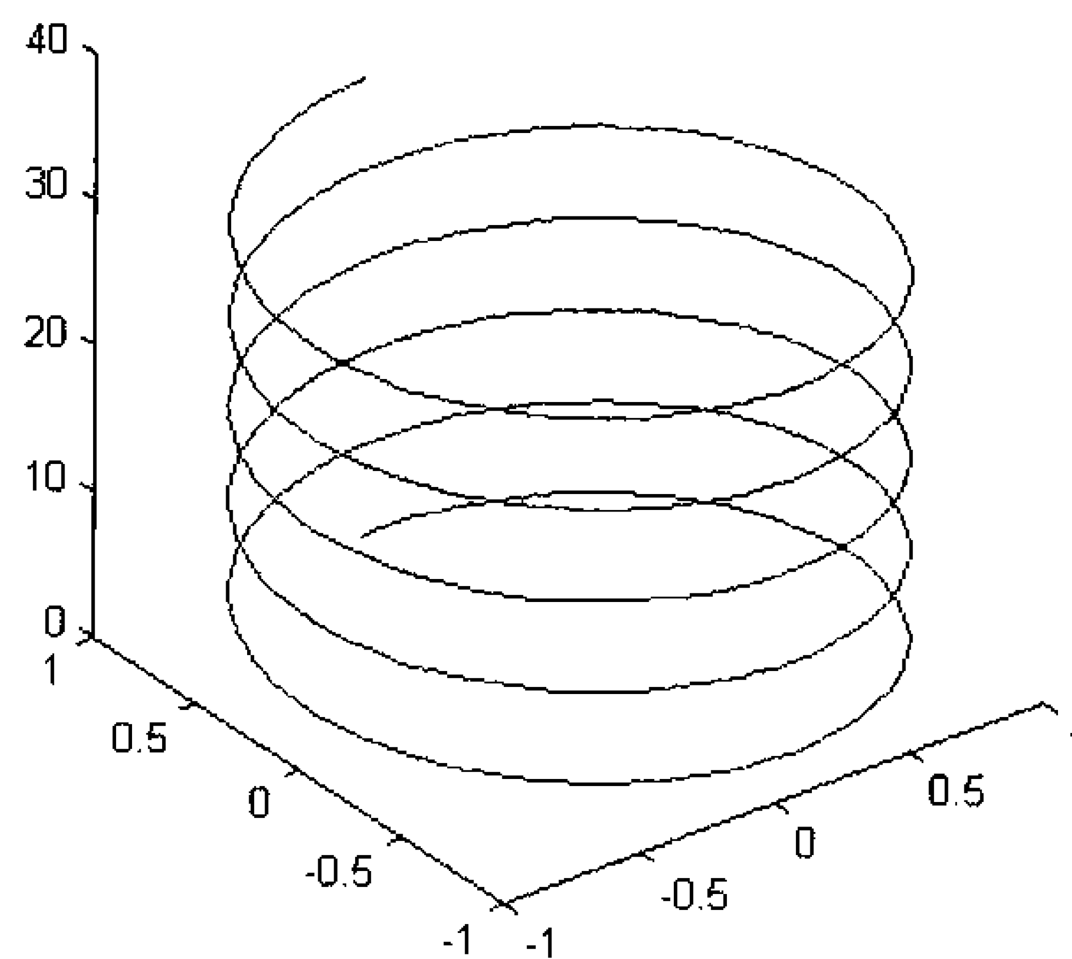


图 4-62 comet3 绘图结果

4.6.2 以旋转颜色映像的方式呈现动画

此种产生动画的方式问题较多，因此不建议使用。它必须通过 spinmap 函数来产生动画，调用格式如表 4-54 所示。

表 4-54　spinmap 函数调用格式

调用格式	说　明
spinmap	旋转颜色映像 3 秒，以颜色变换来产生动画
spinmap(T)	旋转颜色映像 *T* 秒
spinmap(inf)	无限旋转颜色映像直到按下【Ctrl+C】组合键
spinmap(T,inc)	使用定义的间隔秒数与总秒数来旋转颜色映像

下面的命令将 peaks 函数产生的图形由 spinmap 函数生成动画，产生类似波浪的效果。

```
>> peaks
>> spinmap(inf,2)
```

不过此种方式有一个非常大的缺点，那就是系统屏幕的色彩品质只能应用于 256 色，其他（如 16 位、32 位）都无法使用，并且可能在某种操作系统下无法使用，所以要看用户的设置值而定，也就是说，使用此种做法所建立的动画可能会因为系统设置的不同而使动画无法运作。

4.6.3　以电影播放的方式呈现动画

顾名思义，就是先保存多幅不同的图片（欲产生动画的图片），然后存储成一系列各种类型的二维或三维图，再像放电影一样把它们按次序播放出来。这种操作方式首先必须由 getframe 函数将当前的图片抓取作为电影的画面后（将每个欲播放的画面抓取后，以行向量的存储方式置于电影矩阵 *M* 中），再由 movie 函数一次将动画放映出来；另外，也可以使用 movie 来指定播放的次数，如 movie（*M*，20），表示重复播放 20 次。电影方式的结构如下所示。

```
%记录电影
for j=1:n    %旋转并记录每一个画面
    plot_command    %以绘图函数来产生动画
    M(j)=getframe;    %抓取画面值
end
movie(M)    %播放动画
```

以下为建立一个绕 *Z* 旋转的 peaks 动画范例，这个范例主要是应用坐标轴视角的改变来产生动画。程序如下所示。

例程 4.62　利用坐标轴视角产生动画

```
[X,Y,Z]=peaks(30);
surf(X,Y,Z);
set(gca,'visible','off');
colormap(hot)
shading interp
%记录电影
for i=1:15    %旋转并记录每一个画面
        view(-45+15*(i-1),30)    %视角的改变
        m(:,i)=getframe;    %抓取画面值并加到绘图窗口的画面矩阵中
end
movie(m)    %播放画面
```

读者由以上的结果会发现，使用电影产生动画就是将画面依次呈现出来，因为它必须先存储画面，所以，最大的缺点就是要相当大的内存。

4.6.4　以对象的方式呈现动画

此种方式是通过 MATLAB 句柄式图形搭配 drawnow 来实现的，原理是以对象的更新来产生新图，进而覆盖旧图，使图形对象不断发生变化，以实现动画效果，因此曲线、坐标轴等图形对象都可以借助 xdata、ydata、zdata 等属性的变化，搭配 drawnow 函数，来控制图形对象产生动画的效果，不过对于比较复杂的动画在实行上可能比较难以达到。使用此种方式产生动画必须先了解擦除模式（EraseMode）的相关属性。EraseMode 属性主要是用以控制显示与擦除线条对象的技术，因此用户必须了解自己的动画适合哪一种擦除模式，使动画能够呈现最佳的显示方式。

以对象方式呈现动画的步骤如下。

（1）产生一个图形对象，如曲线。

（2）设置该对象的属性 EraseMode 来决定对象更新的方式，一般习惯设置为 xor。

（3）建立循环并借助对象的 xdata 或 ydata 或二者的搭配 drawnow 来产生动画。

以下为建立一个随时间衰减的正弦曲线动画范例，这个范例主要是使用 set 函数来改变曲线的 Y 坐标，并以 xor 的方式抹去旧曲线。

例程 4.63　正弦曲线动画

```
t=0:0.05:10*pi;
%产生曲线并用 xor 方式抹去旧曲线
h=plot(t,sin(2*t).*exp(-t/5),'EraseMode','xor');
set(gcf,'Position',[450,350,350,250])
for i=1:200
        y=sin(2*t+i/10).*exp(-t/5);
        set(h,'ydata',y);    %不断地更新 y 值
        drawnow;
end
```

程序运行结果如图 4-63 所示。

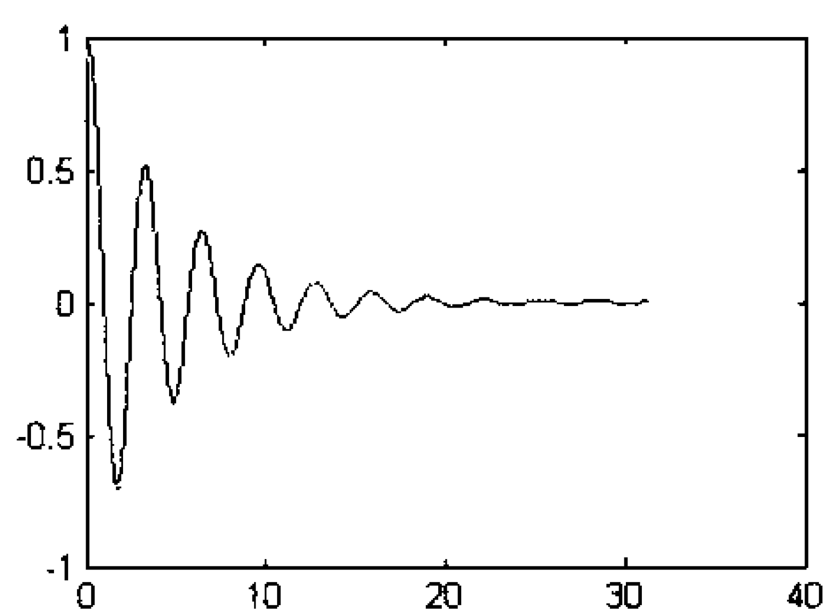


图 4-63　以对象方式呈现动画

用户可以自行将 EraseMode 改成 background 或 none，这样就可以产生不同的动画效果，如将上例的 EraseMode 改成 none，表示旧曲线会被保留并且产生动画。其余使用对象产生动画的范例，用户可以自行参考 MATLAB 内的 lorenz、truss、travel、fitdemo 和 xphide 这几个例子。

```
>>lorenz
```

注意在产生动画效果时，如果该动画是以循环的方式来进行的，切记必须将绘图窗口的 DoubleBuffer 属性设置为 on，如此可以避免动画产生闪烁。如以下语句是将当前窗口的 DoubleBuffer 属性设置为 on 的方式。

```
>>set(gcf,'DoubleBuffer','on')
```

再介绍一个对象呈现动画的控制方式，在这个范例中可以控制 EraseMode 属性为不同值，以便比较出其间的差异，大致来说，虽然 normal 所产生的图形最为精确，并且动画最为美观，但由于速度过慢，无法呈现出连贯性的效果；background 通过将对象颜色变为背景的颜色达到目的，但会使轨迹形成一道空的内容，以致无法呈现出完整的动画；none 则是最差的动画效果，因为它不做任何的擦除动作，因此凡是走过的痕迹都会保留下来，使动画变的很杂乱，所以这个范例还是使用 xor。

例程 4.64　绕轨迹圆球运动示例

```
speed=1000;    %speed 用以控制圆球的速度，值越大速度越慢
%定义轨迹路线
x=linspace(0,2*pi,speed);
y=tan(sin(x))-sin(tan(x));
plot(x,y);
%利用 line 建立球体
%因为还无法知道球体运动的位置，因此使用 line 而不是 plot
%并不需要指定坐标，然后通过该线对象的句柄值 line_handle 来进行球体控制
n=length(x);
line_handle=line('LineStyle','o','LineWidth',5,...
        'MarkerSize',25,'EraseMode','xor',...
        'MarkerEdgeColor','b','MarkerFaceColor','r');
i=1;
%将循环设为全开的
while 1
        set(line_handle,'XDATA',x(i),'YDATA',y(i));    %指定球体的位置
        drawnow    %执行动画
        i=i+1;
        if i>n    %确定该球体能够周而复始地运动
            i=1;
        end
end
```

执行此动画，就会产生一个绕着轨迹跑的圆球运动，用户可以自行测试。得到的结果如图 4-64 所示。

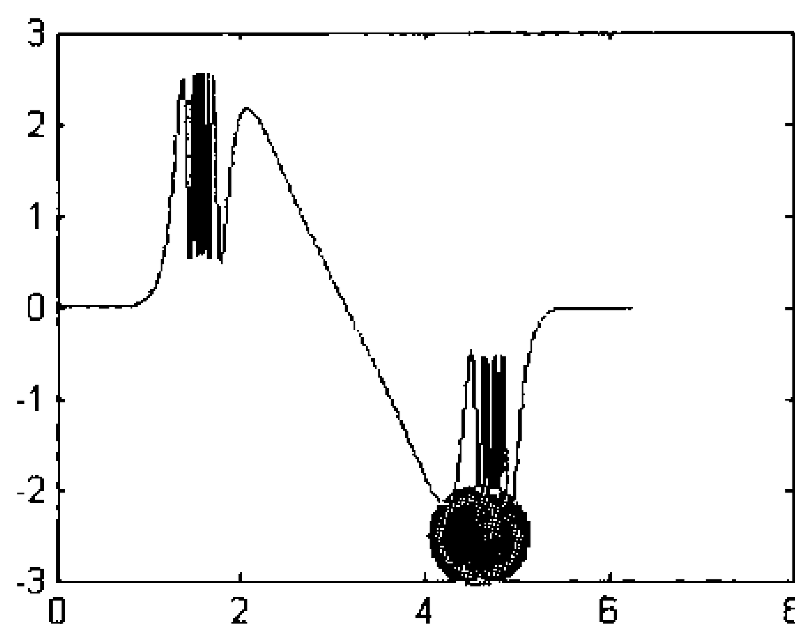


图 4-64　绕轨迹圆球运动

第 5 章　科学计算与应用

在研究与解决工程实际问题中，往往会遇到各种各样的数学计算，这些计算常常难以用手工精确而快捷地进行，必须借助于计算机编制相应的程序做近似计算。MATLAB 为解决此类问题提供了一个很好的计算平台，同时提供了相当丰富的数学函数，用于解决各种实际数学计算问题。

在工程实际中，很多数学问题利用解析方法难以求解，这时就需要借助科学计算方法。本章简单介绍了若干数学问题的数值解法原理，重点介绍了如何利用 MATLAB 提供的函数工具来解决这些问题。

本章主要内容：

- 插值与拟合
- 数值积分和数值微分
- 求解线性方程组
- 求解非线性方程组
- 特征值问题
- 求解常微分方程
- 求解偏微分方程
- 最优化问题

5.1　插值与拟合

在生产和科学实验中，我们往往只掌握有限的测试数据，例如，$y=f(x)$，在区间$[a,b]$上，其中 i=1，2，3，…，n。对于在区间上的其他数据，只能进行估计。这种在已知数据中，用较简单的插值函数 $\phi(x)$通过所有样本点，并对临近数据进行估值计算称为插值。

插值函数 $\phi(x)$必须通过所有样本点。然而在有些情况下，样本点的取得本身就包含着实验中的测量误差，这一要求无疑是保留了这些测量误差的影响，满足这一要求虽然使样本点处“误差”为零，但会使非样本点处的误差变得过大，很不合理。为此，提出了另一种函数逼近方法——数据拟合法，它不要求构造的近似函数 $\phi(x)$全部通过样本点，而是“很好逼近”它们。

插值与拟合在生产和科学实验中，都有着广泛的应用。MATLAB 提供了进行插值与拟合运算的函数，可以方便地进行插值与拟合运算。

5.1.1　一维插值问题

一维插值是进行数据分析的重要手段，MATLAB 提供了 interp1()函数进行一维多项式

插值。interp1()函数使用多项式技术，用多项式函数通过所提供的数据点，并计算目标插值点上的插值函数值，其调用格式如下。

（1）（YI=interp1(X,Y,XI,’method’)

对数据向量 X 和 Y 依选用的方法构造插值函数，并计算 XI 处的函数值，返回给 YI。method 指定插值方法，其选项如下：

’nearest’——最邻近插值。

’linear’——线性插值，为默认设置。

’cubic’——三次插值。

’spline’——三次样条插值。

这几种方法在速度、平滑性、内存使用方面有所区别，在使用时可以根据需要进行选择，包括：

①最邻近法插值是最快的方法，但是，利用它得到的结果平滑性最差。

②线性插值要比最邻近插值占用更多的内存，运行时间略长。与最邻近法不同，它生成的结果是连续的，但是在顶点处会有坡度变化。

③三次插值需要更多内存，而且运行时间比最邻近法和线性插值要长。但是，使用此方法时，插值数据及其导数都是连续的。

④三次样条插值的运行时间相对来说最长，内存消耗比三次插值略少。它生成的结果平滑性最好。但是，如果输入数据不很均匀，可能会得到意想不到的结果。

所有的插值方法要求 X 的元素是单调的，可不等距。当 X 的元素是单调、等距时，使用’*linear’、’*nearest’、’*cubic’或’*spline’选项可快速得到插值结果。如果 Y 是矩阵，那么 Y 的各列将以 X 为公共的横坐标，计算多个（等于 Y 的列数，size(Y,2)）插值函数，输出值 YI 将是 XI 维数×size(Y,2)矩阵。超出范围[Xmin,Xmax]的 XI 值，YI 将返回 NaN。

（2）YI=interp1(Y,XI)

这里，X 和 method 均为默认设置，即 X=1:N，其中 N=size(Y);method=linear。

例 5.1　已知的数据点来自函数 ，根据生成的数据进行插值处理，得出较平滑的曲线直接生成数据。

解：先绘制样本点图，在 MATLAB 命令窗口输入：

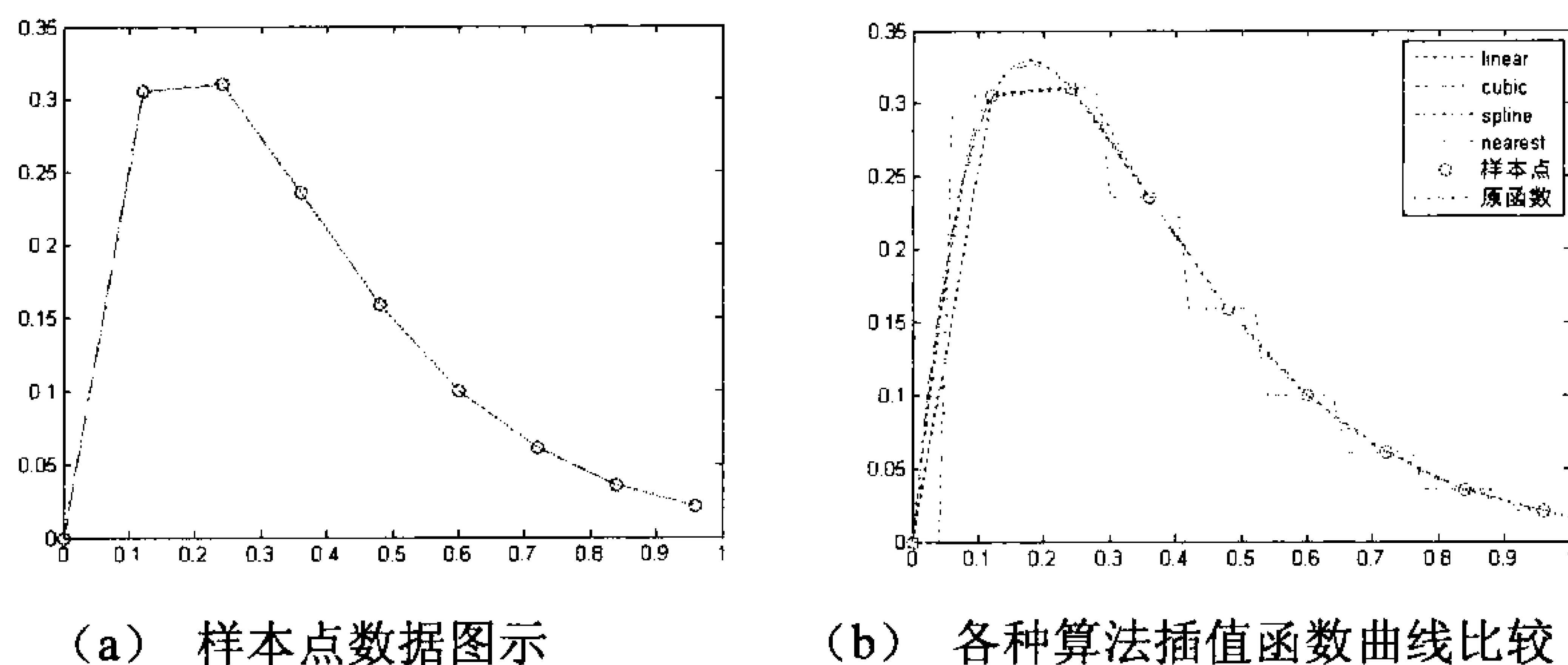
```
>> x=0:.12:1;
>> y=(x.^2-3*x+5).*exp(-5*x).*sin(x); %等距输入样本点
>> plot(x,y,'o',x,y) %绘制样本点（已知数据点），如图 5-1（a）所示。
```

可以看出，由这样的数据直接连线绘制出的曲线十分粗糙，可以再选择一组插值点，然后直接调用 interp1()函数进行插值。

```
>> x1=0:.02:1; %要插值点
>> y0=(x1.^2-3*x1+5).*exp(-5*x1).*sin(x1);
>> y1=interp1(x,y,x1); %默认为线性插值
>> y2=interp1(x,y,x1,'cubic'); %三次 Hermite 插值
>> y3=interp1(x,y,x1,'spline'); %三次样条插值
>> y4=interp1(x,y,x1,'nearest'); %最临近插值
>> plot(x1,[y1' y2' y3' y4'],':',x,y,'o',x1,y0) %绘图比较各插值方法计算结果
>> legend('linear','cubic','spline','nearest','样本点','原函数')
>> [max(abs(y0-y1)),max(abs(y0-y2)),max(abs(y0-y3)),max(abs(y0-y4))] %计算各插值方法最大计算误差
```

```
ans =
        0.0614    0.0177    0.0086    0.1598
```

分别选择各种插值方法，可以得出插值函数曲线与理论曲线，它们之间的比较如图 5-1（b）所示。



（a） 样本点数据图示　　　（b） 各种算法插值函数曲线比较

图 5-1　插值函数曲线图

可以看出，默认的线性插值得到的曲线和图 5-1（a）中同样粗糙，因为该方法就是对各个样本点的直线连接，而'nearest'方法的插值结果就更差了。采用'cubic'和'spline'方法的插值函数曲线更接近于理论值。事实上，应用样条插值算法得出的插值结果十分逼近理论值，甚至用肉眼难以分辨。所以样条插值在一维数据插值中还是很有效的。

MATLAB 还提供了一个快速一维插值函数，其调用格式如下：

```
YI=interp1q(X,Y,XI)
```

当 X 数据不等距时，interp1q 比 interp1 快，但当 X 数据等距时，不如使用'*linear'选项的 interp1 快。interp1q 的使用与 interp1 类似，这里不再重复。

5.1.2　二维插值问题

二维插值是对两个自变量的插值。二维插值在图像处理和数据可视化方面有着非常重要的应用。MATLAB 提供了两个函数 interp2 和 griddata 来实现此功能。其中 interp2 函数用于对二维网格数据进行插值；griddata 函数用于二维随机数据点的插值。

1．二维网格数据插值

MATLAB 提供了二维网格数据插值函数 interp2，其调用格式如下。

（1）ZI=interp2(X,Y,Z,XI,YI)

矩阵 X 和 Y 指定 2-D 区域数据点，在这些数据点处数值矩阵 Z 已知，依此构造插值函数 Z=F(X,Y)，返回在相应数据点 XI、YI 处函数值 ZI=F(XI,YI)。对超出范围[Xmin,Xmax,Ymin,Ymax]的 XI 和 YI 值将返回 ZI=NaN。

（2）ZI=interp2(Z,XI,YI)

这里默认的设置为 X=1:N,Y=1:M，其中，[M,N]=size(Z)。即 N 为矩阵 Z 的行数，Y 为矩阵 Z 的列数。

（3）ZI=interp2(…,'method')

这里 method 是指下列方法之一：

'nearest'——最邻近插值。

'linear'——双线性插值，为默认设置。

'cubic'——双三次插值。

'spline'——样条插值。

所有插值方法要求 X 和 Y 的元素是单调的，即单调递增或单调递减，可不等距。当 X 和 Y 的元素是单调等距时，使用'*linear'、'*cubic'、'*nearest'或'*spline'选项可快速得到插值结果。对一元向量 XI 和 YI，应先使用指令[XI,YI]=meshgrid(xi,yi)生成数据点矩阵 XI 和 YI。

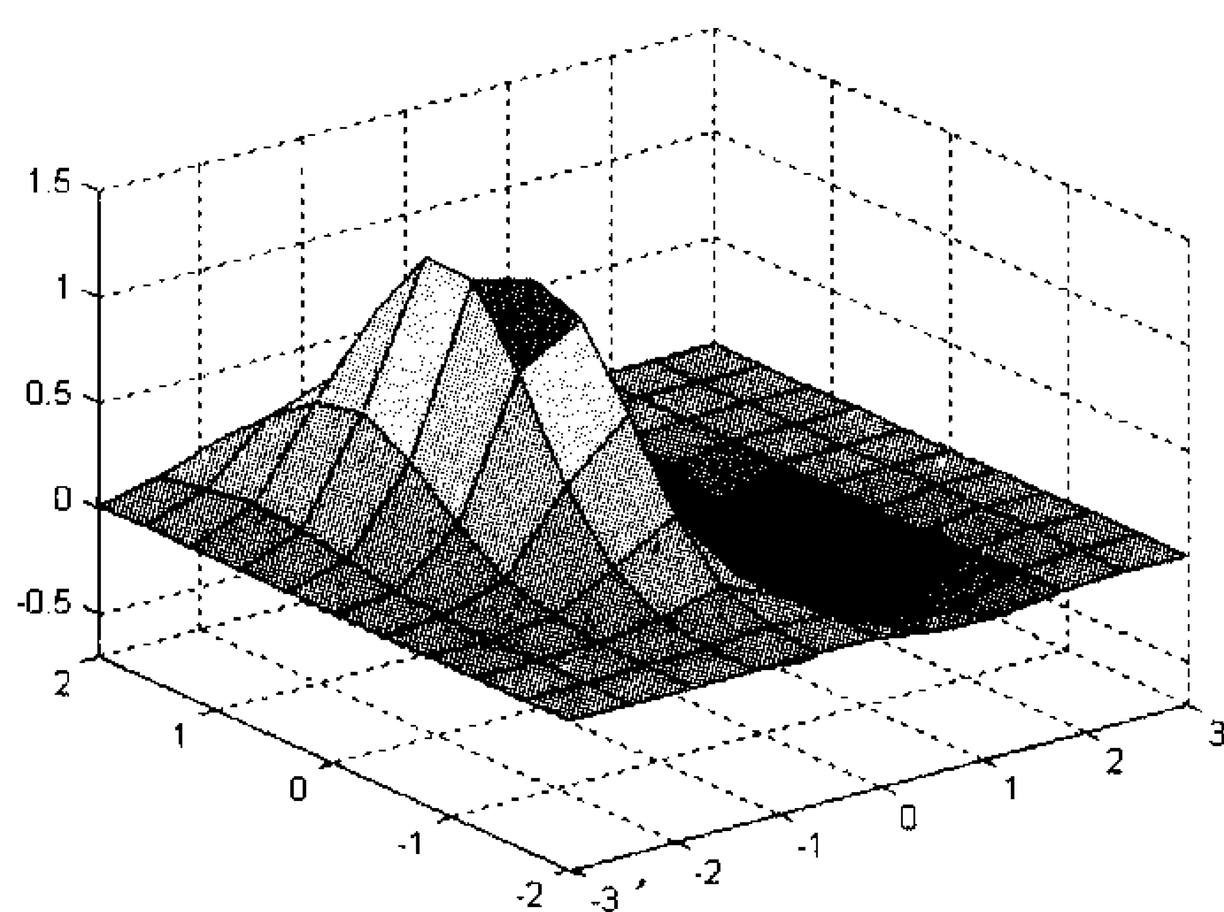
例 5.2 由 $z=f(x,y)=(x^2-2x)\mathrm{e}^{-x^2-y^2-xy}$ 可计算出一些较稀疏的网格数据，对整个函数曲面进行各种插值拟合，并比较插值结果。

解：绘制已知数据的网格图，在 MATLAB 命令窗口输入：

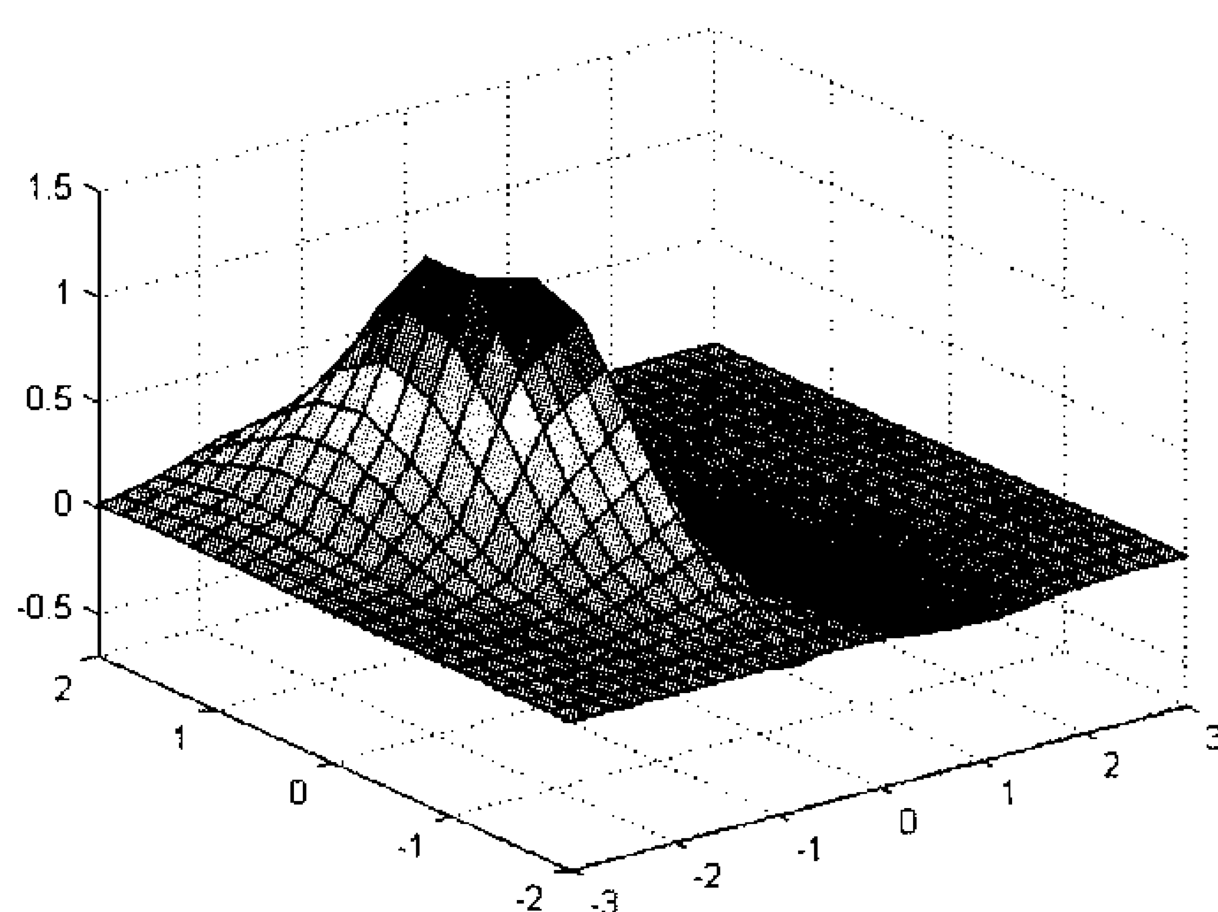
```
>> [x,y]=meshgrid(-3:.6:3,-2:.4:2);
>> z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
>> surf(x,y,z),axis([-3,3,-2,2,-0.7,1.5])
```

选择较密的插值点，则可以用下面的 MATLAB 语句采用默认的插值算法进行插值，得出的结果如图 5-2（b）所示。

```
>> [x1,y1]=meshgrid(-3:.2:3,-2:.2:2);
>> z1=interp2(x,y,z,x1,y1);
>> figure;
>> surf(x1,y1,z1),axis([-3,3,-2,2,-0.7,1.5])
```



（a）已知数据的图示

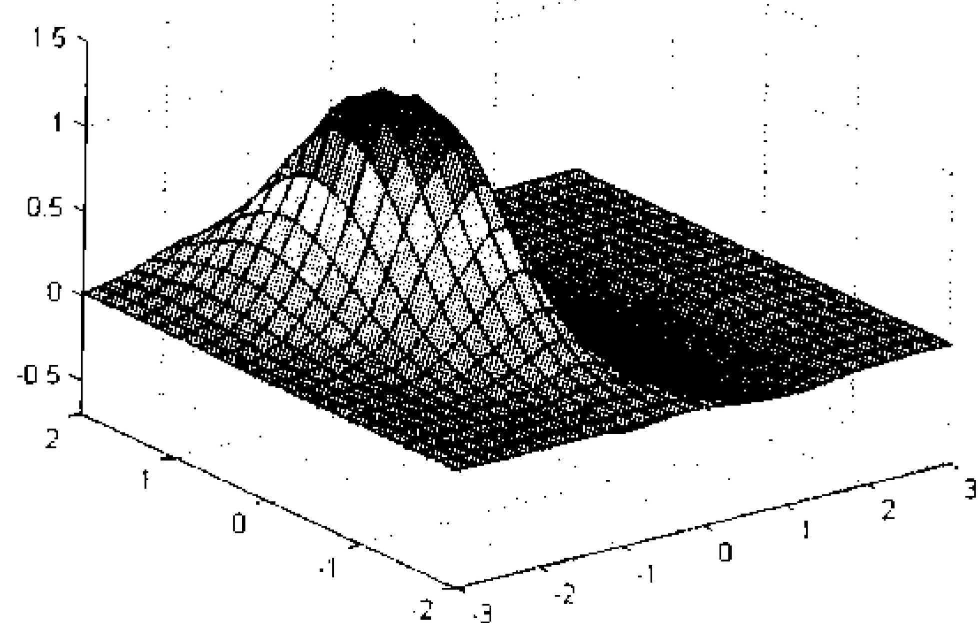


（b）线性插值结果

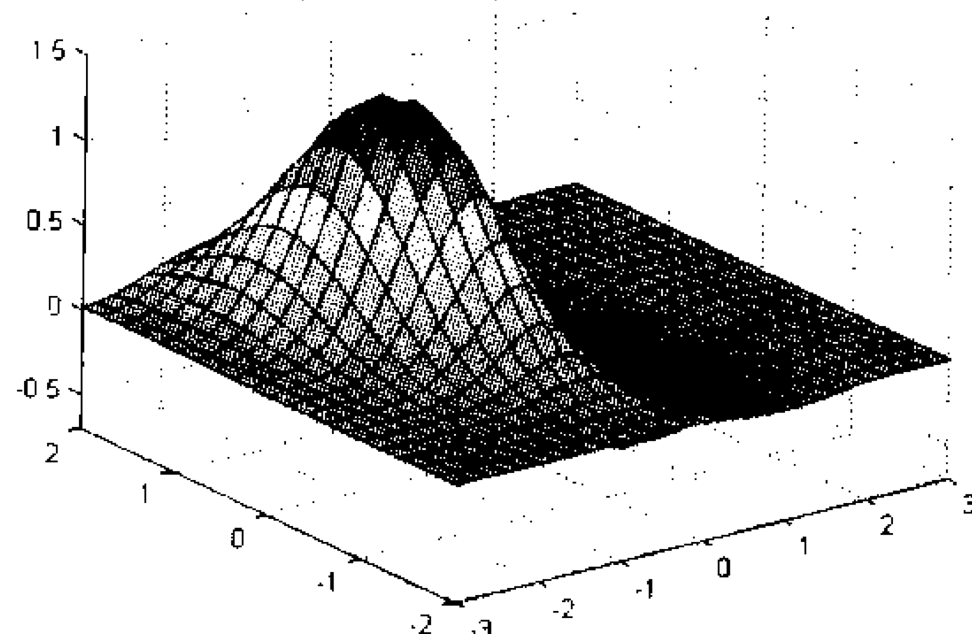
图 5-2　二维函数插值比较

可以看出，默认的线性插值方法还原后的三维表面图在很多地方还是很粗糙。可以用下面的命令分别由立方插值选项和样条插值选项来进行插值，得出的结果如图 5-3 所示。

```
>> z2=interp2(x,y,z,x1,y1,'cubic');
>> figure;
>> surf(x1,y1,z2),axis([-3,3,-2,2,-0.7,1.5])
>> z3=interp2(x,y,z,x1,y1,'spline');
>> figure;
>> surf(x1,y1,z3),axis([-3,3,-2,2,-0.7,1.5])
```



（a）立方插值算法



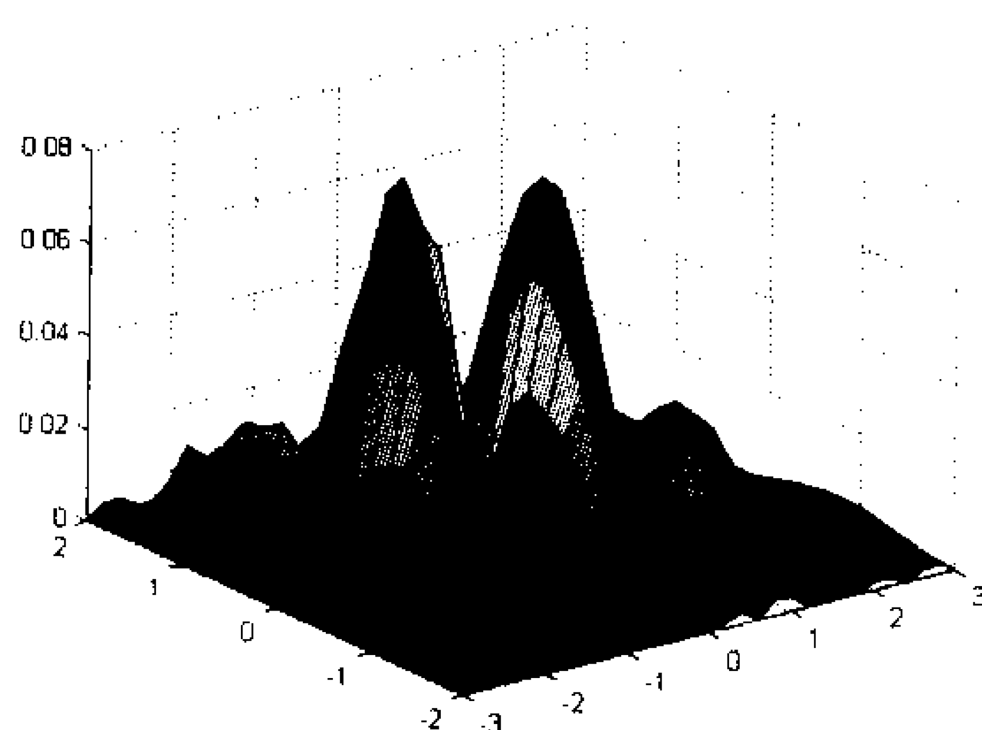
（b）样条插值算法

图 5-3 二维函数其他插值结果比较

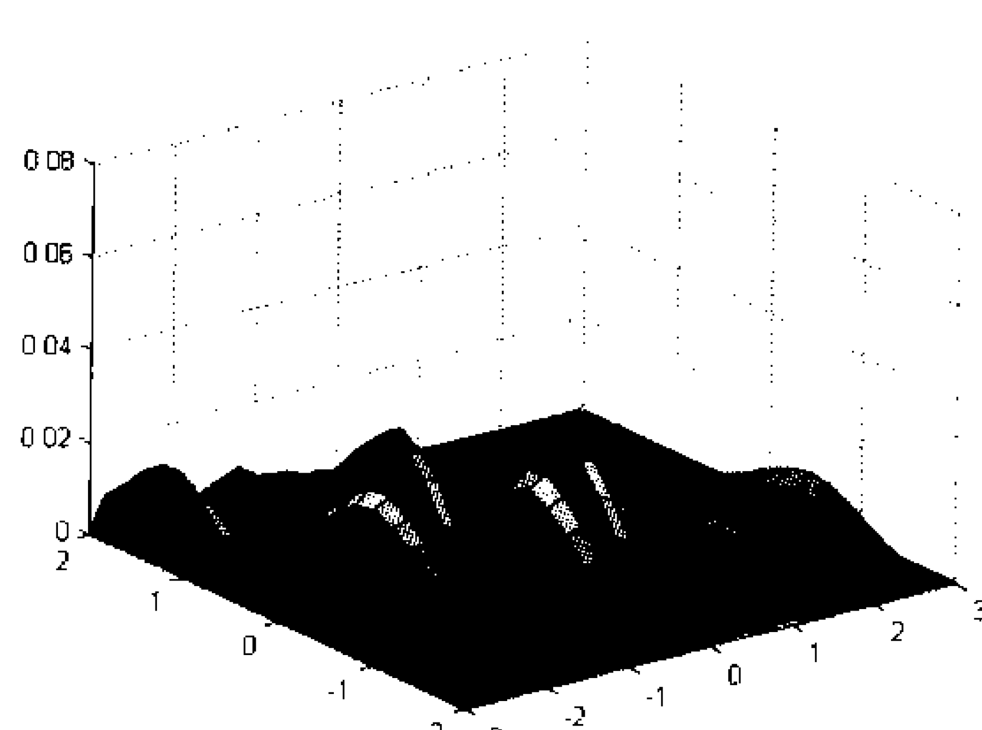
可以看出，这样的插值结果还都是比较理想的。

通过下面的误差分析，可以对'cubic'和'spline'两种插值方法作进一步比较。因为网格已知，故可以由已知函数计算出 z 的精确值 z0，可以通过下面的语句求出两种算法得出的矩阵 z2 和 z3 与真值 z0 之间误差的绝对值，分别如图 5-4（a）和图 5-4（b）所示。可以看出，选择样条方法的插值精度要远高于立方插值算法，所以在实际应用中建议选用'spline'插值选项。

```
>> z0=(x1.^2-2*x1).*exp(-x1.^2-y1.^2-x1.*y1);
>> figure;
>> surf(x1,y1,abs(z0-z2)),axis([-3,3,-2,2,0,0.08])
>> figure;
>> surf(x1,y1,abs(z0-z3)),axis([-3,3,-2,2,0,0.08])
```



（a）立方插值算法误差图示



（b）样条插值算法误差图示

图 5-4 二维函数的误差

2．二维随机数据点的插值

通过上面的例子可以看出，interp2 函数能够较好地进行二维插值运算。但该函数有一个重要缺陷，就是它只能处理以网格形式给出的数据。如果已知数据不是以网格形式给出的，则该函数是无能为力的。在实际应用中，大部分问题都是以实测的多组(x_i, y_i, z_i)点给出的，所以不能直接使用函数 interp2 进行二维插值。

MATLAB 提供了一个更一般的 griddata()函数，用来专门解决这样的问题。其调用格式如下。

（1）ZI=griddata(X,Y,Z,XI,YI)

其中，X，Y，Z 是已知的样本点坐标，这里并不要求是网格型的，可以是任意分布的，均由向量给出。XI、YI 是期望的插值位置，可以是单个点，可以是向量或网格型矩阵，得出的 ZI 应该维数和 XI、YI 一致，表示插值的结果。

（2）[XI,YI,ZI]=griddata(X,Y,Z,XI,YI)

这里[XI,YI]=meshgrid(XI,YI)

（3）[…]=griddata(…,'method')

这里 method 是指下列方法之一：

'linear'——基于三角剖分的线性插值，为默认设置。

'cubic'——基于三角剖分的三次插值。

'nearest'——最邻近插值。

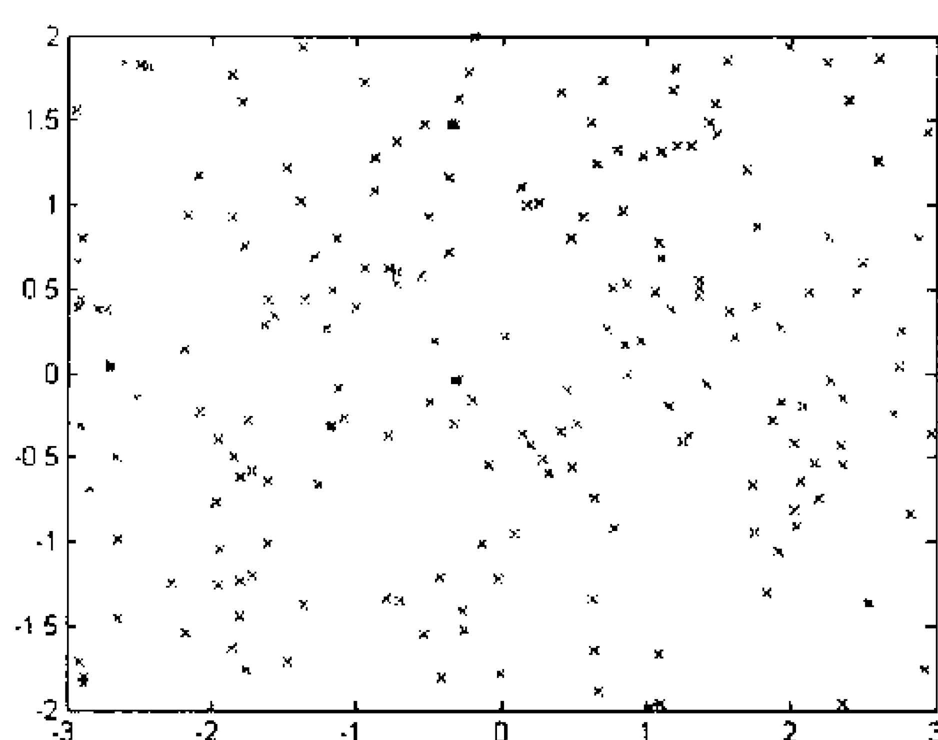
'v4'——MATLAB 4.0 版本中提供的插值算法。

当方法'linear'和'nearest'生成曲面函数分别有不连续一阶导数和不连续时，'cubic'和'v4'方法将生成光滑曲面。除'v4'外所有的方法都基于数据的 Delaunay 三角剖分。

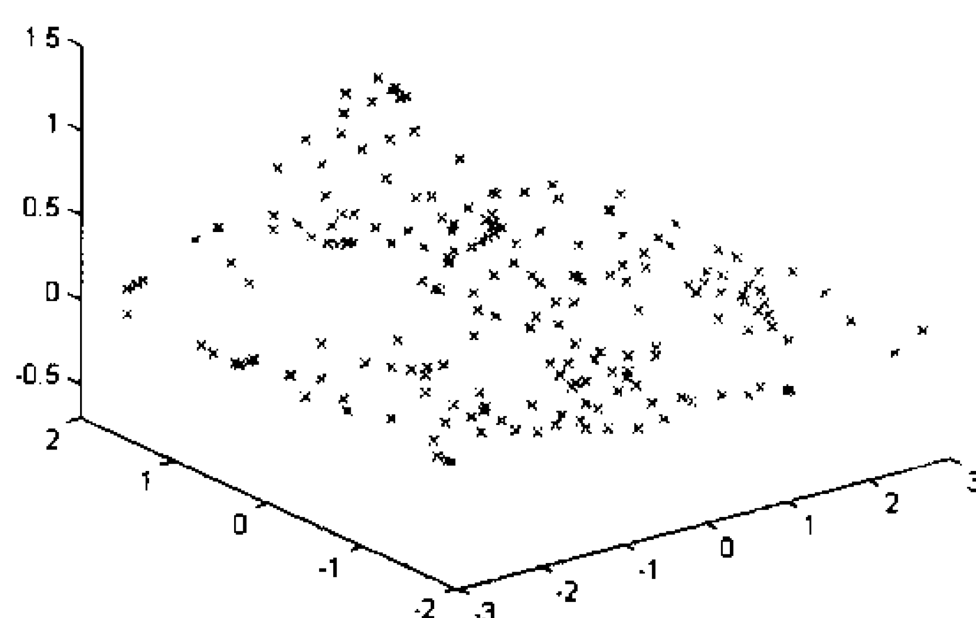
例 5.3 已知二元函数 $z=f(x,y)=(x^2-2x)e^{-x^2-y^2-xy}$，在 $x\in[-3,3]$，$y\in[-2,2]$ 矩形区域内随机选择一组 (x_i,y_i) 坐标，就可以生成一组 z_i 的值。以这些值为已知数据，用一般分布数据插值函数 griddata()进行插值处理，并进行误差分析。

解：这里选择 200 个随机数构成的点，则可以用下面的语句生成 x，y，z 向量，但由于这些数据不是网格数据，所以得出的数据向量不能直接用三维曲面的形式表示。但可以用下面的语句将各个样本点在 x-y 平面上的分布形式显示出来，如图 5-5（a）所示，也可以绘制出样本点的三维分布，如图 5-5（b）所示。可以看出，这些分布点是比较随机的。

```
>> x=-3+6*rand(200,1);y=-2+4*rand(200,1);
>> z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y); %生成已知数据
>> plot(x,y,'x') %样本点的二维分布
>> figure,plot3(x,y,z,'x'),axis([-3,3,-2,2,-0.7,1.5]) %样本点的三维分布
```



（a）已知数据点的分布



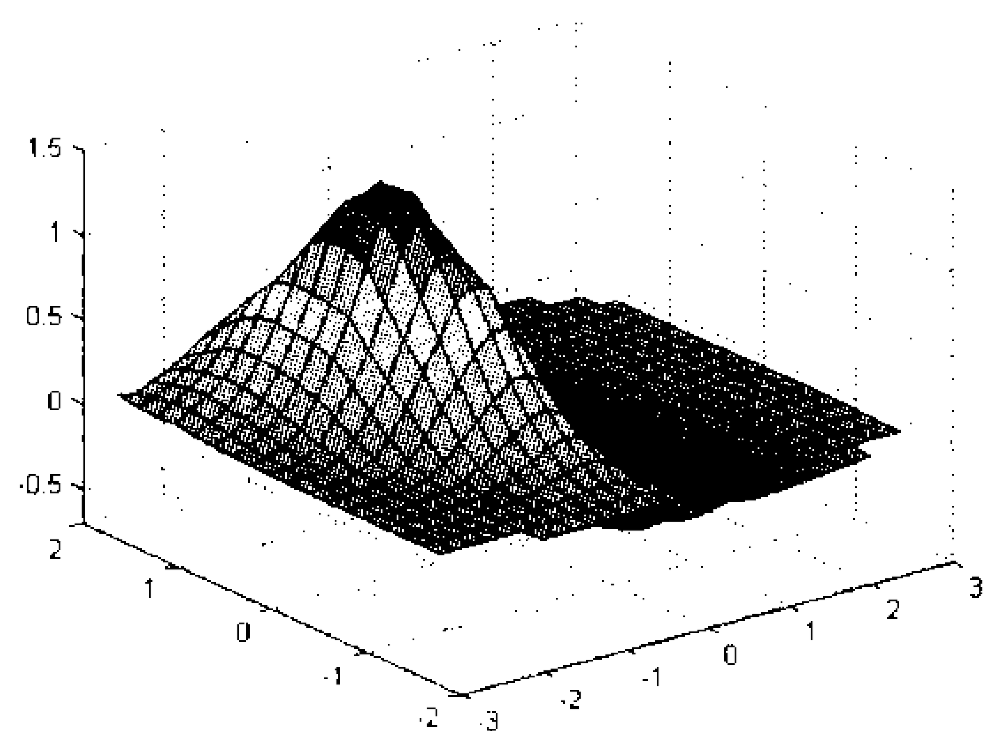
（b）已知数据点的三维分布

图 5-5 已知样本数据显示

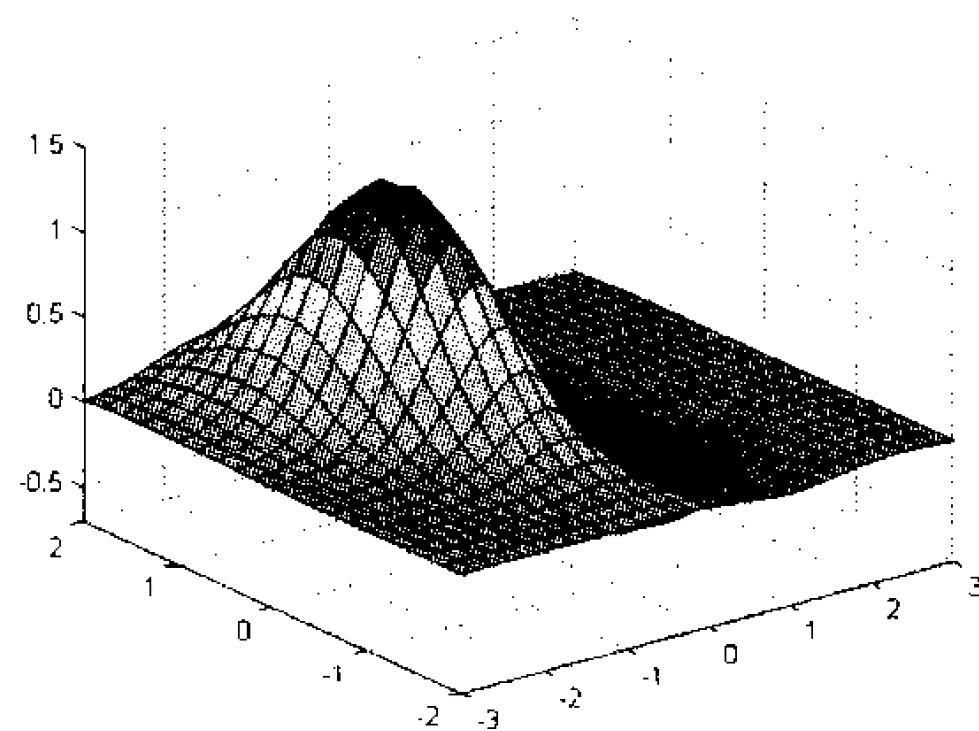
用下面的语句生成网格矩阵作为插值点，用'cubic'和'v4'两种算法获得插值结果，还可以绘制出拟合后的曲面形式，分别如图 5-6（a）和图 5-6（b）所示。可以看出，用'v4'算法得出的结果效果明显更好些。

```
>> [x1,y1]=meshgrid(-3:.2:3,-2:.2:2);
```

```
>> z1=griddata(x,y,z,x1,y1,'cubic');
>> surf(x1,y1,z1),axis([-3,3,-2,2,-0.7,1.5])
>> z2=griddata(x,y,z,x1,y1,'v4');
>> figure,surf(x1,y1,z2),axis([-3,3,-2,2,-0.7,1.5])
```



（a）立方插值算法

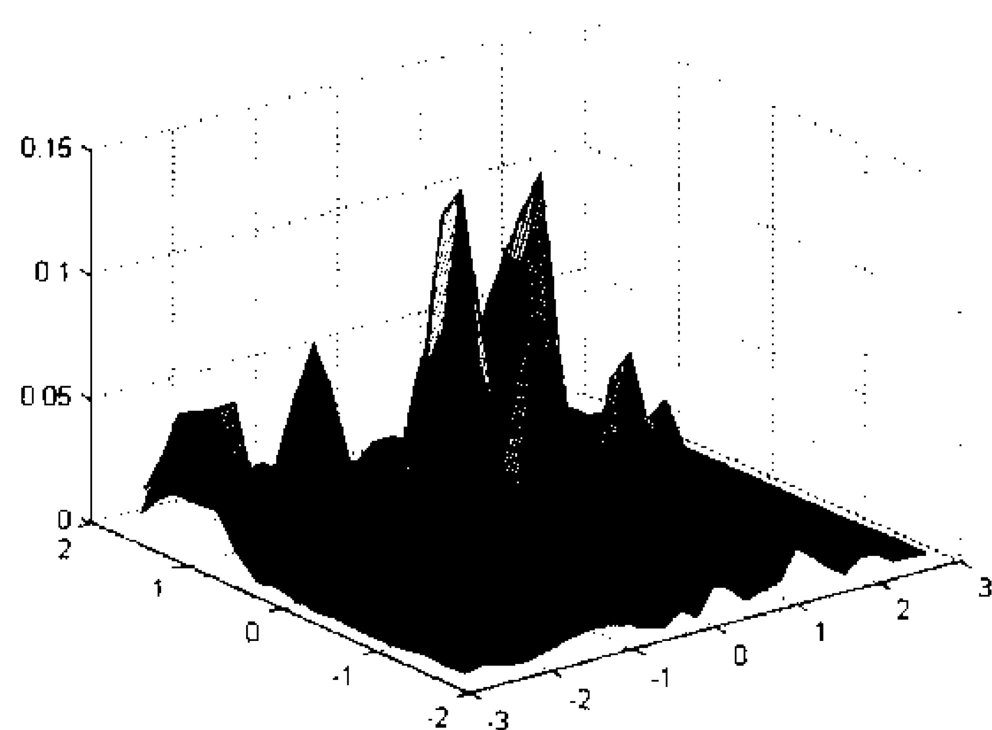


（b）'v4'插值算法

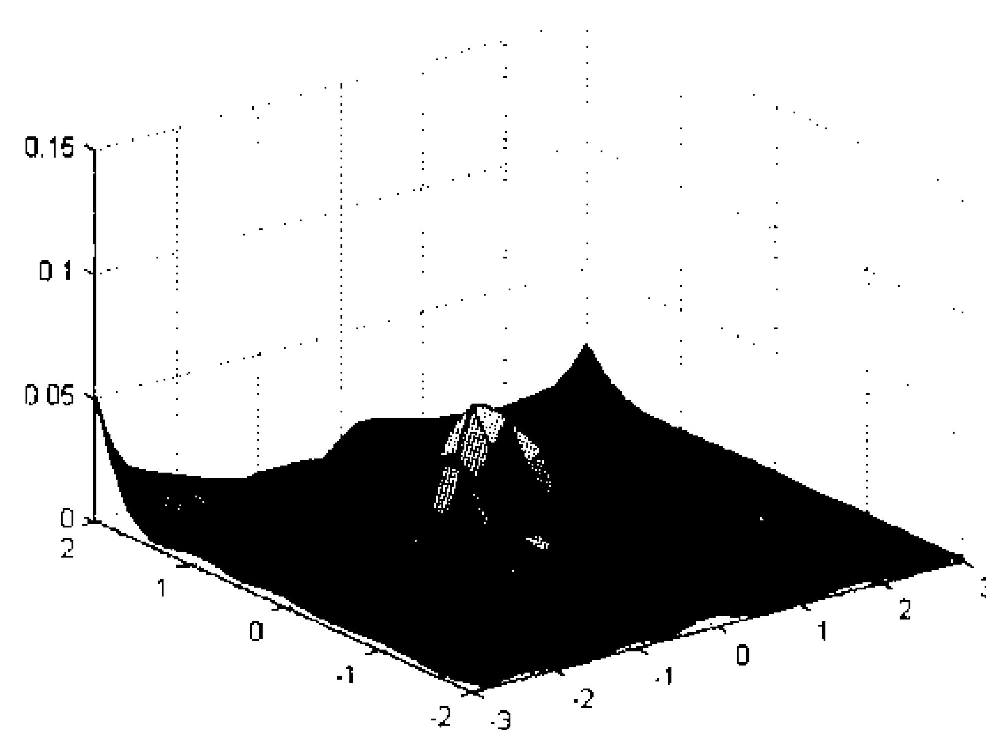
图 5-6 二维函数各种插值结果比较

还可以进一步进行误差分析。用下面的语句可以先计算出在新网格点处函数值的精确解，并用这些点和两种方法计算出来的误差，得出如图 5-7（a）、图 5-7（b）所示的误差曲面。可见，用'v4'选项的插值结果明显优于立方插值算法，所以在实际应用中建议采用该算法。

```
>> z0=(x1.^2-2*x1).*exp(-x1.^2-y1.^2-x1.*y1); %新网格各点的函数值
>> surf(x1,y1,abs(z0-z1));axis([-3,3,-2,2,0,0.15])
>> figure ,surf(x1,y1,abs(z0-z2));axis([-3,3,-2,2,0,0.15])
```



（a）立方插值算法



（b）'v4'插值算法

图 5-7 二维函数各种插值误差比较

5.1.3 样条插值

有些情况下，插值函数不仅要求保证各段曲线在连接点上的连续性，还要求确保整条曲线在这些点上的光滑性。例如，在船体、飞机等外型曲线的设计中，不仅要求曲线连续，也要求曲率连续，这就要求插值函数具有连续的二阶导数。样条插值可以满足上述要求。

在 interp1 等插值指令的'method'选项中提供了样条插值'spline'选项，可以用来进行三

次样条插值。除此之外 MATLAB 系统中还提供了 spline 等指令用来进行三次样条插值。其调用格式如下：

（1）*YY*=spline(*X*,*Y*,*XX*)　根据给定数据点(*X*,*Y*)应用三次样条插值方法计算 *XX* 所对应的值 *YY*。

（2）*PP*=spline(X,Y)　根据给定数据点(*X*,*Y*)获得三次样条函数数据，供 PPVAL 等指令使用。

（3）*YY*=PPVAL(*PP*,*XX*)　根据逐段多项式样条函数数据 PP（如由 PP=spline(*X*,*Y*)指令得到的），计算 *XX* 对应插值函数值 *YY*。

X 必须是向量。*Y* 可以是向量或矩阵。当 *Y* 为向量时，*Y* 必须和 *X* 具有相同的维数，且 *X*(*j*)与 *Y*(*j*)——对应，*j*=1，2，…当 *Y* 为矩阵时，每一列作为一个向量，即 *X*(*j*)与 *Y*(:, *j*)相对应。

对于更专业、更丰富的样条插值函数，可参考 MATLAB 的样条工具箱。

5.1.4　曲线拟合

插值函数 $\phi(x)$必须通过所有样本点。然而，在有些情况下，样本点的取得本身就包含着实验中的测量误差，这一要求无疑是保留了这些测量误差的影响，满足这一要求虽然使样本点处"误差"为零，但会使非样本点处的误差变得过大，很不合理。为此，提出了另一种函数逼近方法——数据拟合法，它不要求构造的近似函数 $\phi(x)$全部通过样本点，而是"很好逼近"它们。常用的数据拟合方法有多项式拟合和最小二乘拟合。

1．多项式拟合

一般多项式拟合的目标是找出一组多项式系数 $a_i, i=1$，2，…，$n+1$，使得多项式：$\psi(x)=a_1x^n+a_2x^{n-1}+\cdots+a_nx+a_{n+1}$能够较好地拟合原始数据。和前面介绍的插值算法不同，多项式拟合并不能保证每个样本点都在拟合的曲线上，但能使得整体的拟合误差较小。多项式拟合可以通过 MATLAB 提供的 polyfit()函数实现。该函数的调用格式如下：

```
p=polyfit(x,y,n)
```

其中，*x* 和 *y* 为原始的样本点构成的向量，*n* 为选定的多项式阶次，得出的 *p* 为多项式系数按降幂排列得出的行向量，可以用符号运算工具箱中的 poly2sym()函数将其转化成多项式形式，也可以使用 polyval()函数求取多项式的值。下面将通过例子演示多项式拟合函数的使用方法和优缺点。

例 5.4　考虑例 5.1 中的样本点数据，试用多项式拟合的方法在不同的阶次下进行拟合，并观察拟合效果，找出合适的阶次。

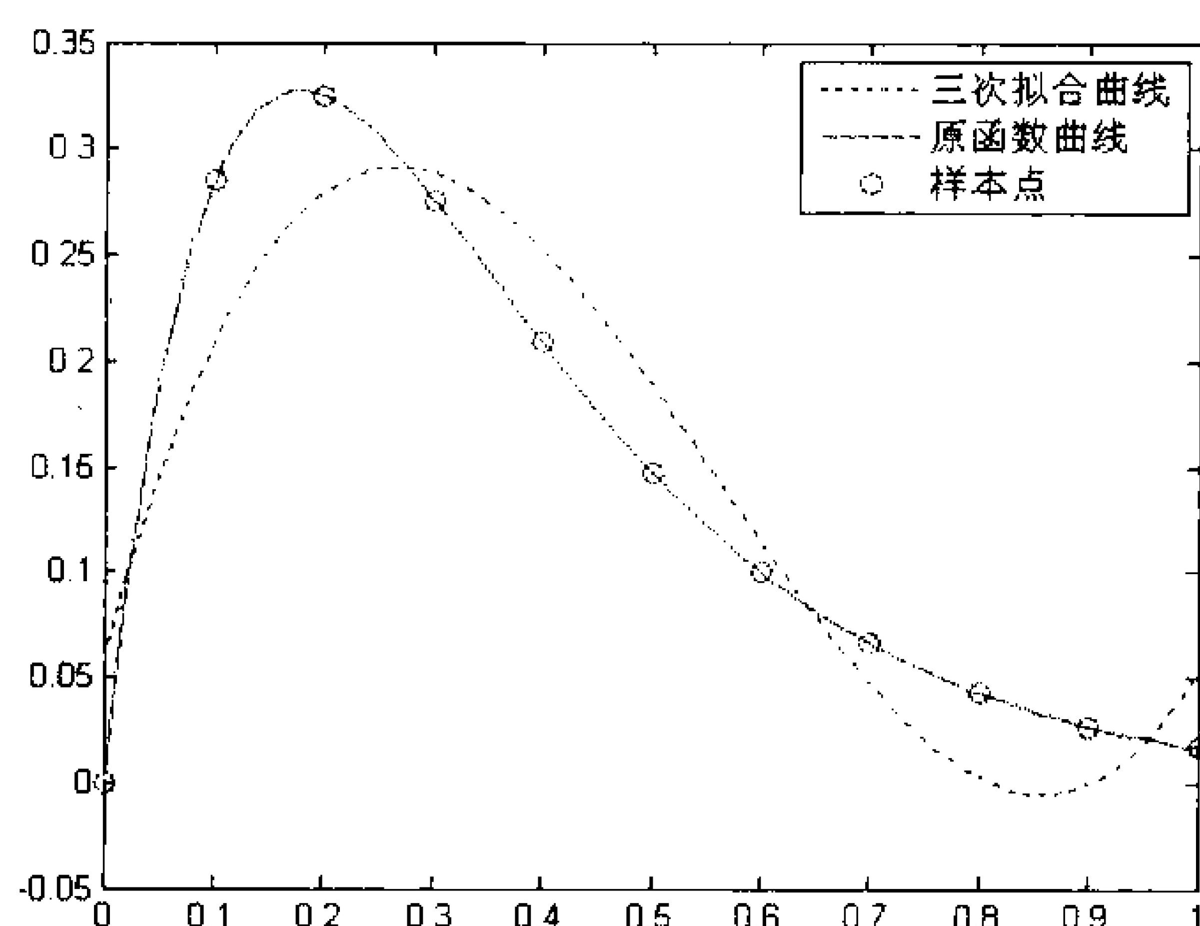
解：可以用下面的语句得出拟合该数据的三次多项式并绘制出拟合曲线，如图 5-8（a）所示。

```
>> x0=0:.1:1;y0=(x0.^2-3*x0+5).*exp(-5*x0).*sin(x0);
>> p3=polyfit(x0,y0,3);vpa(poly2sym(p3),10)   %可以如下显示多项式
ans =
    2.839962923*x^3-4.789842696*x^2+1.943211631*x+.5975248921e-1
>> x=0:.01:1;ya=(x.^2-3*x+5).*exp(-5*x).*sin(x);
>> y1=polyval(p3,x);
```

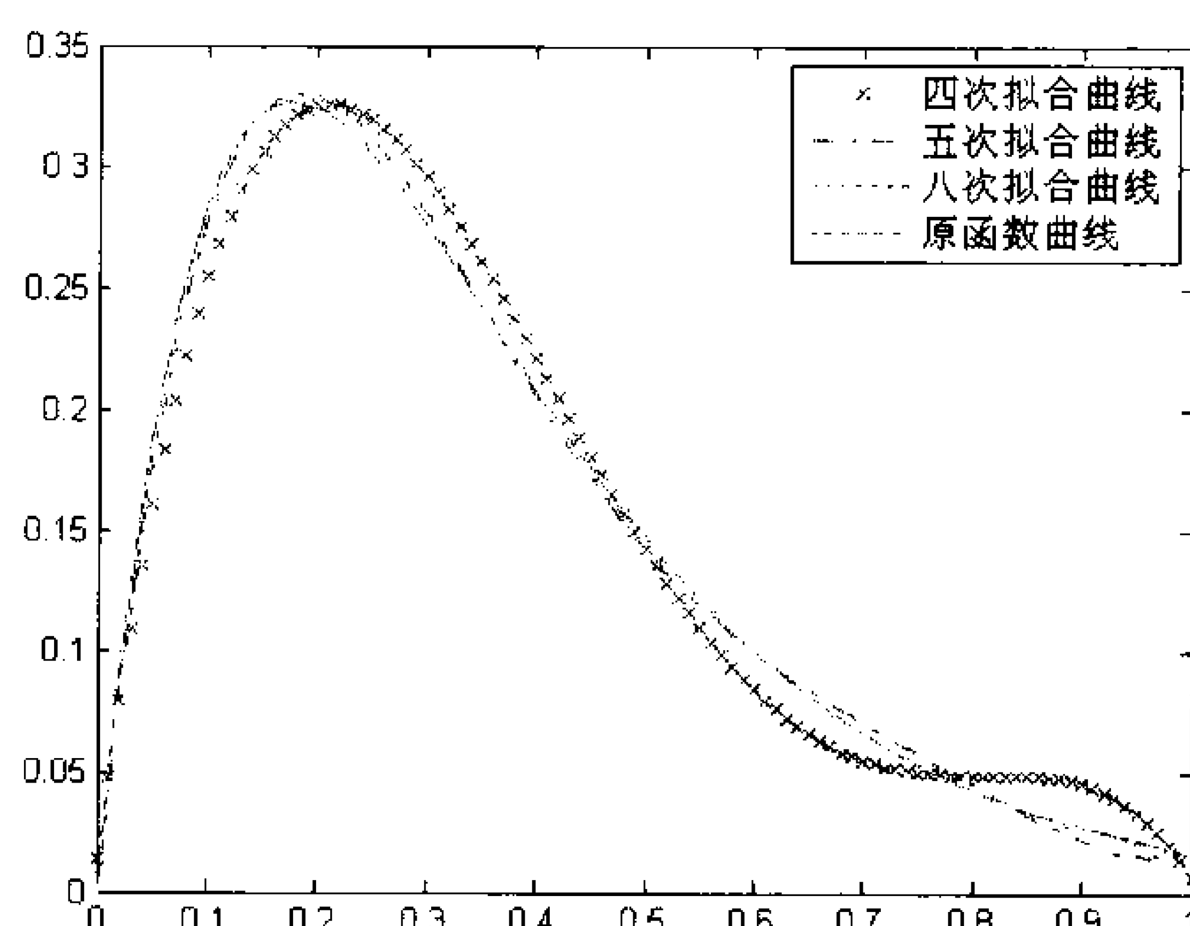
```
>> plot(x,y1,x,ya,x0,y0,'o'),legend('三次拟合曲线','原函数曲线','样本点')
```

从拟合结果可以看出，结果还是相当差的。一种很显然的解决方法就是提高拟合多项式的次数，下面就不同的次数进行拟合，最终得出如图 5-8（b）所示的拟合效果。

```
>> p4=polyfit(x0,y0,4);y4=polyval(p4,x);
>> p5=polyfit(x0,y0,5);y5=polyval(p5,x);
>> p8=polyfit(x0,y0,8);y8=polyval(p8,x);
>> plot(x,y4,'x',x,y5,'-.',x,y8,':',x,ya,'-')
>> legend('四次拟合曲线','五次拟合曲线','八次拟合曲线','原函数曲线')
```



（a） 三次多项式拟合



（b） 其他次数的多项式拟合

图 5-8 多项式拟合效果

从该例的拟合效果看，当拟合多项式的次数等于 8 时，多项式拟合曲线与原函数曲线已经基本重合，拟合效果已经比较令人满意。这时，拟合多项式可以由下面的语句显示出来，可以用该多项式在[0,1]区间内近似原函数模型。

```
>> vpa(poly2sym(p8),5)
ans =
-8.2586*x^8+43.566*x^7-101.98*x^6+140.22*x^5-125.29*x^4+74.450*x^3-27.672*x^2+
4.9869*x+.42037e-6
```

例 5.5 已知的数据点来自函数 $f(x)=1/(1+25x^2)$，$-1\leqslant x\leqslant 1$，根据生成的数据点进行多项式曲线拟合，观察拟合效果。

解：取不同的多项式阶次，使用如下的语句获得多项式拟合，并绘制出拟合曲线与原函数曲线进行对比，如图 5-9 所示。

```
>> x0=-1+2*[0:10]/10;y0=1./(1+25*x0.^2);
>> x=-1:.01:1;ya=1./(1+25*x.^2);
>> p3=polyfit(x0,y0,3);y3=polyval(p3,x);
>> p5=polyfit(x0,y0,5);y5=polyval(p5,x);
>> p8=polyfit(x0,y0,8);y8=polyval(p8,x);
>> p10=polyfit(x0,y0,10);y10=polyval(p10,x);
>> plot(x,ya,x,y3,':',x,y5,'.',x,y8,'x',x,y10,'-.')
>> legend('原函数','三次拟合','五次拟合','八次拟合','十次拟合')
```

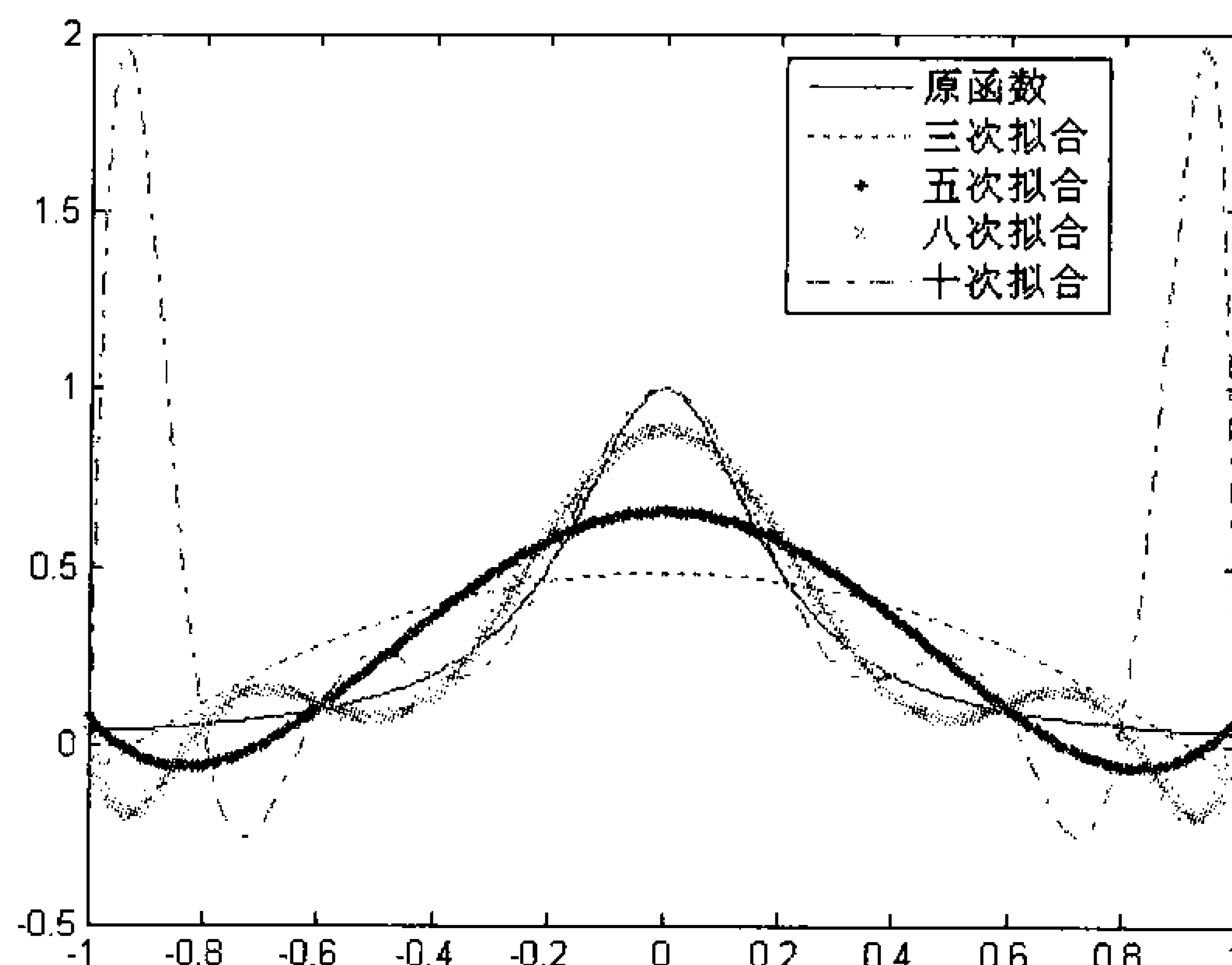


图 5-9　各阶多项式拟合效果

由该例可以看出，多项式拟合并不是阶数越高越好，多项式拟合的效果也并不一定总是很精确，有时结果是相当差的，甚至说是完全错误的。

2．最小二乘拟合

假设有一组数据 x_i，y_i，$i=1$，2，…，N，且已知这组数据满足某一函数原型 $\hat{y}(x)=f(a,x)$，其中 a 是待定系数向量，则最小二乘曲线拟合的目标就是求出这一组待定系数的值，使得目标函数 $J=\min\limits_a\sum\limits_{i=1}^N\left[y_i-\hat{y}(x_i)\right]^2=\min\limits_a\sum\limits_{i=1}^N\left[y_i-f(a,x_i)\right]^2$ 为最小。在 MATLAB 的最优化工具箱中提供了 lsqcurvefit()函数，可以解决最小二乘曲线拟合的问题。该函数的调用格式如下：

$$[a,J_m]=\text{lsqcurvefit(Fun, } a_0, x, y)$$

其中，Fun 为原型函数的 MATLAB 表示，可以是 M-函数或 inline()函数，a_0 为最优化的初值，x,y 为原始输入输出数据向量，调用该函数则将返回待定系数向量 a 以及在此待定系数下的目标函数的值 J_m。

例 5.6　已知数据 (x_i,y_i)，满足 $y_i=0.12\mathrm{e}^{-0.213x_i}+0.54\mathrm{e}^{-0.17x_i}\sin(1.23x_i)$ 其中，x_i=10(i-1)/100，i=1，2，…，101。并已知该数据满足函数原型 $y(x)=a_1\mathrm{e}^{-a_2x}+a_3\mathrm{e}^{-a_4x}\sin(a_5x)$，其中，$a_i$ 为待定系数。采用最小二乘曲线拟合的目的就是获得这些待定系数，使得目标函数的值最小。

解：根据已知的函数原型，可以编写出如下的 MATLAB 函数：

```
>> f=inline('a(1)*exp(-a(2)*x)+a(3)*exp(-a(4)*x).*sin(a(5)*x)','a','x')
```

建立起函数的原型，则可以由下面的语句得出待定系数向量。

```
>> x=0:.1:10;y=0.12*exp(-0.213*x)+0.54*exp(-0.17*x).*sin(1.23*x);
>> [xx,res]=lsqcurvefit(f,[1,1,1,1,1],x,y);xx',res
Optimization terminated: relative function value
 changing by less than OPTIONS.TolFun.
ans =
      0.1197
      0.2125
      0.5404
```

```
        0.1702
        1.2300
res =
        7.1637e-007
```

可以看出，这样得出的待定系数精度较高，接近于理论值 $a=[0.12,0.213,0.54,0.17,1.23]^T$。如果想进一步提高精度，则需要修改最优化的选项，这时函数的调用格式也将发生变化。

```
>> ff=optimset;ff.TolFun=1e-20;ff.TolX=1e-15;%修改精度限制
>> [xx,res]=lsqcurvefit(f,[1,1,1,1,1],x,y,[],[],ff);xx',res
Optimization terminated: relative function value
 changing by less than OPTIONS.TolFun.
ans =
        0.1200
        0.2130
        0.5400
        0.1700
        1.2300
res =
        9.5035e-021
>> x1=0:.01:10;y1=f(xx,x1);plot(x1,y1,x,y,'o')
>> legend('拟合曲线','样本点')
```

其中，两个空矩阵表示 a 向量的上下限，由于对这些参数的范围无限制，故采用了默认的表示形式。可以看出，修改误差后，得出的拟合待定系数更加精确。绘制出的拟合曲线与样本点如图 5-10 所示。

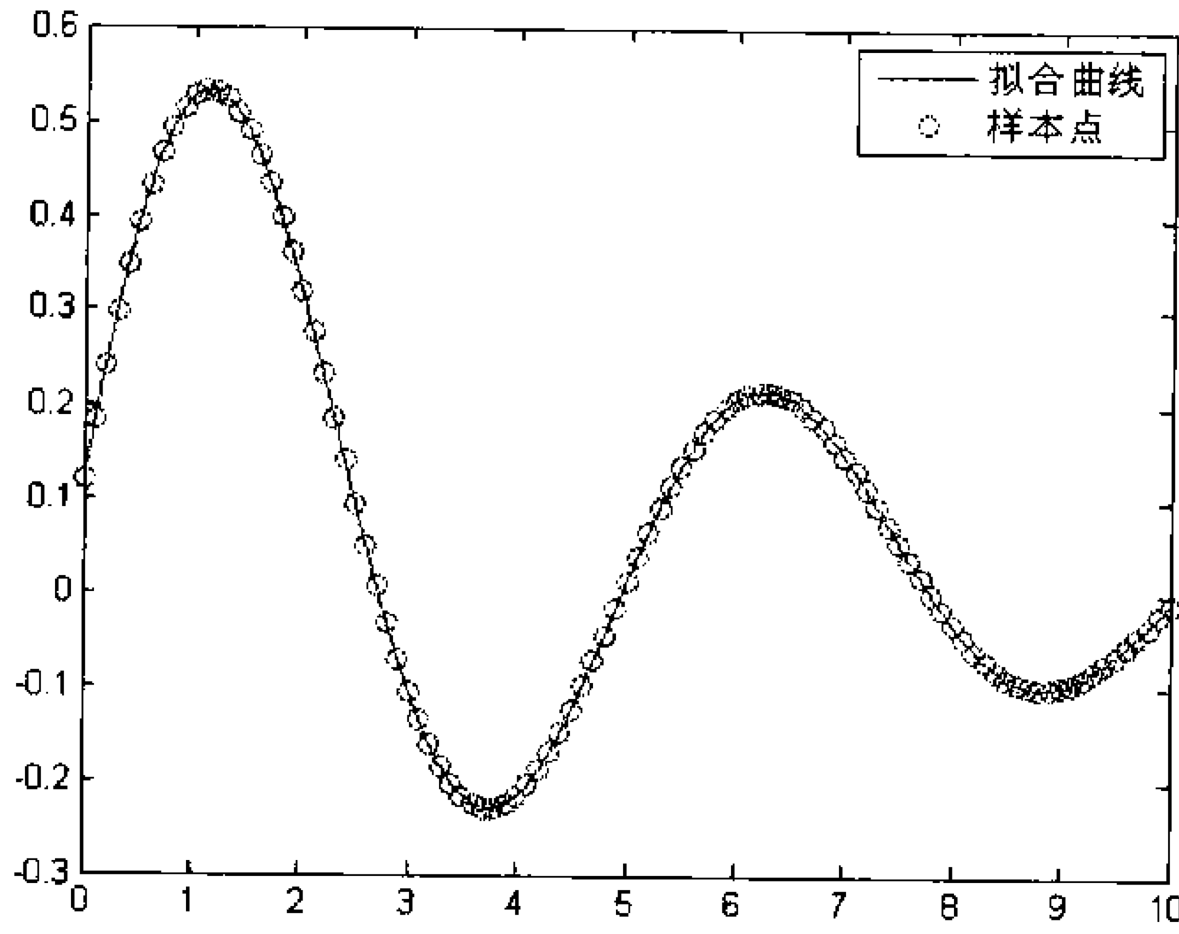


图 5-10　拟合效果比较

例 5.7　假设有如下一组实测数据：

x_i	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y_i	2.3201	2.6470	2.9707	3.2885	3.6008	3.9090	4.2147	4.5191	4.8232	5.1275

假设已知该数据可能满足的原型函数为 $y(x)=ax+bx^2\mathrm{e}^{-cx}+d$，试求出满足该原型函数的最小二乘解 a，b，c，d 的值。

解：下面的语句可以输入已知的参数。

```
>> x=0.1:0.1:1;
>> y=[2.3201,2.6470,2.9707,3.2885,3.6008,3.9090,4.2147,4.5191,4.8232,5.1275];
```

令 $a_1=a$，$a_2=b$，$a_3=c$，$a_4=d$，这样，原型函数可以写成 $y(x)=a_1x+a_2x^2e^{-a_3x}+a_4$，可以用 MATLAB 建立函数原型为 M-函数。

```
function y=c8f3(a,x)
y=a(1)*x+a(2)*x.^2.*exp(-a(3)*x)+a(4);
```

建立起函数的原型，则可以调用 lsqcurvefit()函数求出待定系数向量。

```
>> a=lsqcurvefit(@c8f3,[1,2,2,3],x,y)
a =
    2.4587    2.4489    1.4466    2.0734
```

用下面的语句可以计算出各个点处的值，可以将二者曲线绘制在同一坐标系下，如图 5-11 所示。可见，二者还是很接近的，说明拟合效果较好。

```
>> y1=c8f3(a,x);plot(x,y,'o',x,y1),legend('样本点','拟合曲线')
```

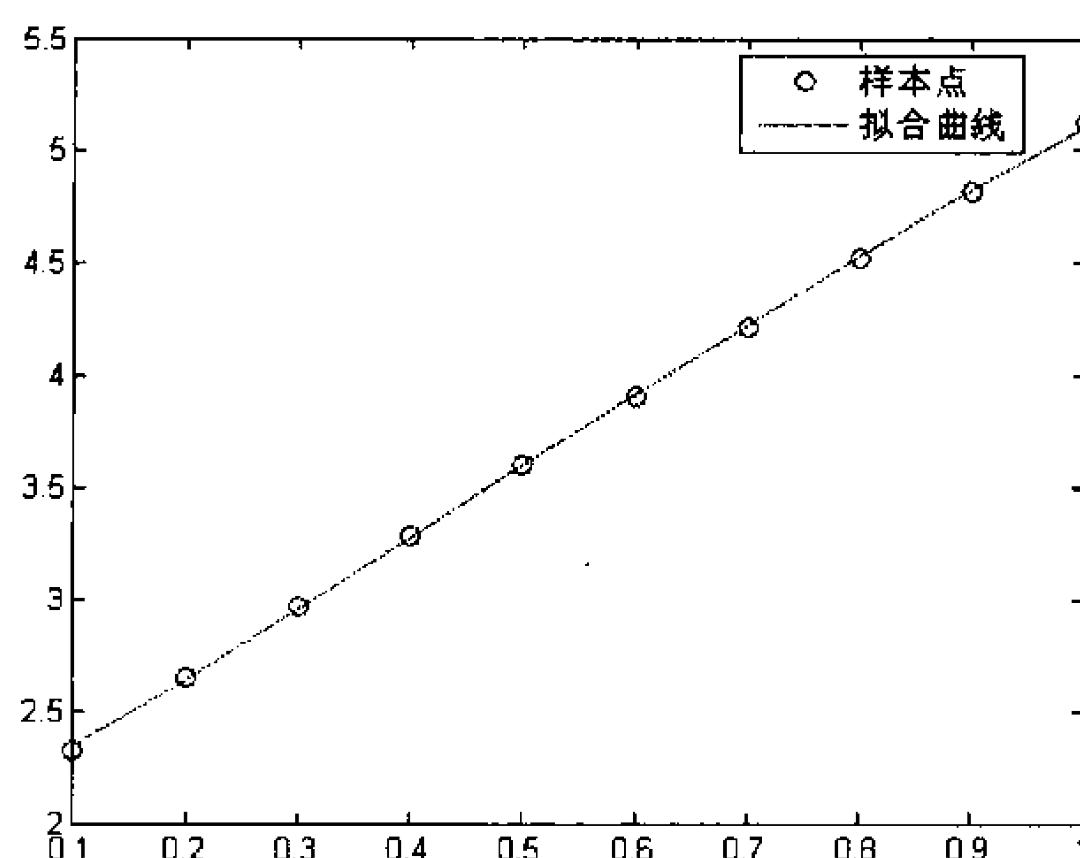


图 5-11　拟合效果图

5.2　数值积分与数值微分

在解决许多科技问题时，经常需要进行定积分的计算。解析计算（即精确计算）积分的方法都是基于 Newton-Leibniz 公式 $\int_a^b f(x)\mathrm{d}x=F(b)-F(a)$ 进行计算。应用该公式必须先求出被积函数 $f(x)$ 的原函数 $F(x)$。但是，很多情况下原函数是很难求得的，或者求得的原函数非常冗长、复杂；有些从科学实验或工程实际中测得的被积函数本身就不是解析表达式，而是表格或曲线，计算它们的定积分，更是无法找到解析形式的原函数。解决该类定积分计算问题，就需要用到数值积分。

5.2.1　数值积分

1．数值积分基本原理

求解定积分的数值方法多种多样，如简单的梯形法、辛普生（Simpson） 法、牛顿—柯特斯(Newton-Cotes)法等都是经常采用的方法。它们的基本思想都是将整个积分区间$[a,b]$分成 n 个子区间$[x_i,x_{i+1}]$，i=1，2，…，n，其中 $x_1=a$，$x_{n+1}=b$。这样求定积分问题就分解为求和问题。

2．数值积分的实现方法

1）变步长辛普生法

基于变步长辛普生法，MATLAB 给出了 quad 函数来求定积分。该函数的调用格式如下：

```
[I,n]=quad('fname',a,b,tol,trace)
```

其中，fname 是被积函数名。a 和 b 分别是定积分的下限和上限。tol 用来控制积分精度，默认时取 tol=0.001。trace 控制是否展现积分过程，若取非 0 则展现积分过程，取 0 则不展现，默认时取 trace=0。返回参数 I 即定积分值，n 为被积函数的调用次数。

例 5.8 求如下函数在区间 $[0,3\pi]$ 内的定积分。

$$f(x)=\mathrm{e}^{-0.5x}\cdot\sin(x+\frac{\pi}{6})$$

解：首先，建立被积函数文件 fesin.m。

```
function f=fesin(x)
f=exp(-0.5*x).*sin(x+pi/6);
```

然后，调用数值积分函数 quad 求定积分。

```
[S,n]=quad('fesin',0,3*pi)
S =
    0.9008
n =
      77
```

2）牛顿—柯特斯法

基于牛顿—柯特斯法，MATLAB 给出了 quadl 函数来求定积分。该函数的调用格式如下：

```
[I,n]=quadl('fname',a,b,tol,trace)
```

其中，参数的含义和 quad 函数相似，只是 tol 的默认值取 10^{-6}。 该函数可以更精确地求出定积分的值，且一般情况下函数调用的步数明显小于 quad 函数，从而保证能以更高的效率求出所需的定积分值。

例 5.9 分别用 quad 函数和 quadl 函数求如下函数定积分的近似值，并在相同的积分精度下，比较函数的调用次数。

$$f(x)=\mathrm{e}^{-x}$$

解：首先，定义被积函数 fx。

```
format long;
fx=inline('exp(-x)');
```

调用函数 quad 求定积分：

```
[I,n]=quad(fx,1,2.5,1e-10)
I =
    0.28579444254766
n =
      65
```

调用函数 quadl 求定积分：

```
[I,n]=quadl(fx,1,2.5,1e-10)
```

```
I =
    0.28579444254754
n =
    33
```

由计算结果可知，利用 quadl 函数进行定积分计算，函数调用的步数明显小于 quad 函数。

3）被积函数由一个表格定义

在 MATLAB 中，对由表格形式定义的函数关系的求定积分问题用 trapz(*X*,*Y*)函数。其中向量 *X*，*Y* 定义函数关系 *Y*=*f*(*X*)。

例 5.10 用 trapz 函数计算定积分。

命令如下：

```
X=1:0.01:2.5;
Y=exp(-X);          %生成函数关系数据向量
trapz(X,Y)
ans =
     0.28579682416393
```

3. 二重定积分的数值求解

使用 MATLAB 提供的 dblquad 函数就可以直接求出二重定积分的数值解。该函数的调用格式如下：

```
I=dblquad(f,a,b,c,d,tol,trace)
```

该函数求 $f(x,y)$在$[a,b]\times[c,d]$区域上的二重定积分。参数 tol，trace 的用法与函数 quad 完全相同。

例 5.11 计算二重定积分。

$$\int_{-2}^{2}\int_{-1}^{1}\mathrm{e}^{-x^2/2}\cdot\sin(x^2+y)\mathrm{d}x\mathrm{d}y$$

解：首先，建立一个函数文件 fxy.m：

```
function f=fxy(x,y)
global ki;
ki=ki+1;                  %ki 用于统计被积函数的调用次数
f=exp(-x.^2/2).*sin(x.^2+y);
```

然后，调用 dblquad 函数求解：

```
global ki;ki=0;
I=dblquad('fxy',-2,2,-1,1)
ki
I =
    1.57449318974494
ki =
    1038
```

5.2.2 数值微分

在实验或工程应用中，有时要根据已知的数据点，求某点的一阶或高阶导数。由于只有离散的数据，没有函数表达式，所以不能用解析的方法进行求导，这时就要用到数

值微分。

MATLAB 语言中没有直接进行数值微分计算的函数，所以本节先介绍数值微分算法，介绍其中较好算法的 MATLAB 实现，最后通过例子演示数值微分程序。

1．**差商与数值微分**

当函数 $f(x)$ 是以离散点列给出时，当函数的表达式过于复杂时，常用数值微分近似计算 $f(x)$ 的导数 $f'(x)$。在微积分中，导数表示函数在某点上的瞬时变化率，它是平均变化率的极限；在几何上可解释为曲线的斜率；在物理上可解释为物体变化的速率。

以下是导数 $f'(x)$ 的 3 种定义形式：

$$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}=\lim_{h\to 0}\frac{f(x)-f(x-h)}{h}=\lim_{h\to 0}\frac{f(x+h)-f(x-h)}{2h} \tag{5.1}$$

在微积分中，用差商的极限定义导数；在数值计算中返璞归真，导数取用差商（平均变化率）作为其近似值。最简单的计算数值微分的方法是用函数的差商近似函数的导数，即取极限的近似值。

用向前差商（平均变化率）近似导数有：

$$f'(x_0)\approx\frac{f(x_0+h)-f(x_0)}{h} \tag{5.2}$$

用向后差商（平均变化率）近似导数有：

$$f'(x_0)\approx\frac{f(x_0)-f(x_0-h)}{h} \tag{5.3}$$

用向前差分和向后差分近似导数，精度都是 $o(h)$ 级的，当 h 稍大时，计算误差会很大。经实践检验，利用基于向前差商和向后差商的数值微分算法求取高阶微分时的精度一般都是较低的。

用中心差商（平均变化率）近似导数有：

$$f'(x_0)\approx\frac{f(x_0+h)-f(x_0-h)}{2h} \tag{5.4}$$

由泰勒展开得中心差商的截断误差：

$$R(x)=f'(x_0)-\frac{f(x_0+h)-f(x_0-h)}{2h}=\frac{h^2}{12}[f'''(\xi_1)+f'''(\xi_2)]=$$

$$\frac{h^2}{6}f'''(\xi)=o(h^2),\quad x_0-h\leqslant\xi\leqslant x_0+h \tag{5.5}$$

可见，中心差商算法的精度为 $o(h^2)$ 级。另外，还有一种中心差商算法，精度为 $o(h^4)$ 级。该中心差商算法的计算公式为：

$$f'(x)=\frac{f(x-2h)-8f(x-h)+8f(x+h)-f(x+2h)}{12h}$$

$$f''(x)=\frac{-f(x-2h)+16f(x-h)-30f(x)+16f(x+h)-f(x+2h)}{12h^2} \tag{5.6}$$

$$f'''(x)=\frac{f(x-3h)-8f(x-2h)+13f(x-h)-13f(x+h)+8f(x+2h)-f(x+3h)}{8h^3}$$

2．数值微分的实现

在 MATLAB 中，没有直接提供求数值导数的函数，只有计算向前差分的函数 diff，其调用格式如下：

DX=diff(X)计算向量 X 的向前差分，DX(i)=X(i+1)－X(i)，i=1，2，…，n-1。

DX=diff(X,n)计算 X 的 n 阶向前差分。例如，diff(X,2)=diff(diff(X))。

DX=diff(A,n,dim)计算矩阵 A 的 n 阶差分，dim=1 时（默认状态），按列计算差分；dim=2 时按行计算差分。

从前面的介绍可知，式（5.6）中给出的微分算法有 $o(h^4)$ 级精度，因而即使 h 不趋向 0 时，仍能得到较好的近似微分。所以这里采用该公式为所选算法，可以编写如下的 MATLAB 函数来进行数值微分的计算。

```
function [dy,dx]=diff_ctr(y,Dt,n)
yx1=[y,0,0,0,0,0];yx2=[0,y,0,0,0,0];yx3=[0,0,y,0,0,0];
yx4=[0,0,0,y,0,0];yx5=[0,0,0,0,y,0];yx6=[0,0,0,0,0,y];
switch n
    case 1
        dy=(-diff(yx1)+7*diff(yx2)+7*diff(yx3)-diff(yx4))/(12*Dt);L0=3;
    case 2
        dy=(-diff(yx1)+15*diff(yx2)-15*diff(yx3)+diff(yx4))/(12*Dt^2);L0=3;
    case 3
        dy=(-diff(yx1)+7*diff(yx2)-6*diff(yx3)-6*diff(yx4)+...
            7*diff(yx5)-diff(yx6))/(8*Dt^3);L0=5;
end
dy=dy(L0+1:end-L0);dx=([1:length(dy)]+L0-2-(n>2))*Dt;
```

这样编写的 M-函数调用格式如下：

```
[dy,dx]=diff_ctr(y,Dt,n)
```

其中，y 为给定的等间距的实测数据构成的向量，Dt 为自变量的间距，n 为所需的导数的阶次（n 不大于 3）。向量 dy 为得出的导数向量，而 dx 为相应的自变量向量。注意这两个向量的长度比 y 短。

数值微分的计算除了可以用差商近似外，还可以对离散数据先进行曲线拟合，然后对拟合得到的函数表达式求导；还可以借助三次样条插值进行三次样条求导。限于篇幅，此处不进行讲述。

例 5.12 用不同的方法求函数 $f(x)$ 的数值导数，并在同一个坐标系中作出 $f'(x)$ 的图像。

程序如下：

```
f=inline('sqrt(x.^3+2*x.^2-x+12)+(x+5).^(1/6)+5*x+2');
g=inline('(3*x.^2+4*x-1)./sqrt(x.^3+2*x.^2-x+12)/2+1/6./(x+5).^(5/6)+5');
x=-3:0.01:3;
p=polyfit(x,f(x),5);            %用 5 次多项式 p 拟合 f(x)
dp=polyder(p);                  %对拟合多项式 p 求导数 dp
dpx=polyval(dp,x);              %求 dp 在假设点的函数值
dx= diff_ctr(f(x),0.01,1);      %利用编写的 diff_ctr()函数对 f(x)求数值导数
gx=g(x);                        %求函数 f 的导函数 g 在假设点的导数
plot(x,dpx,x,dx,'.',x,gx,'-');  %作图
```

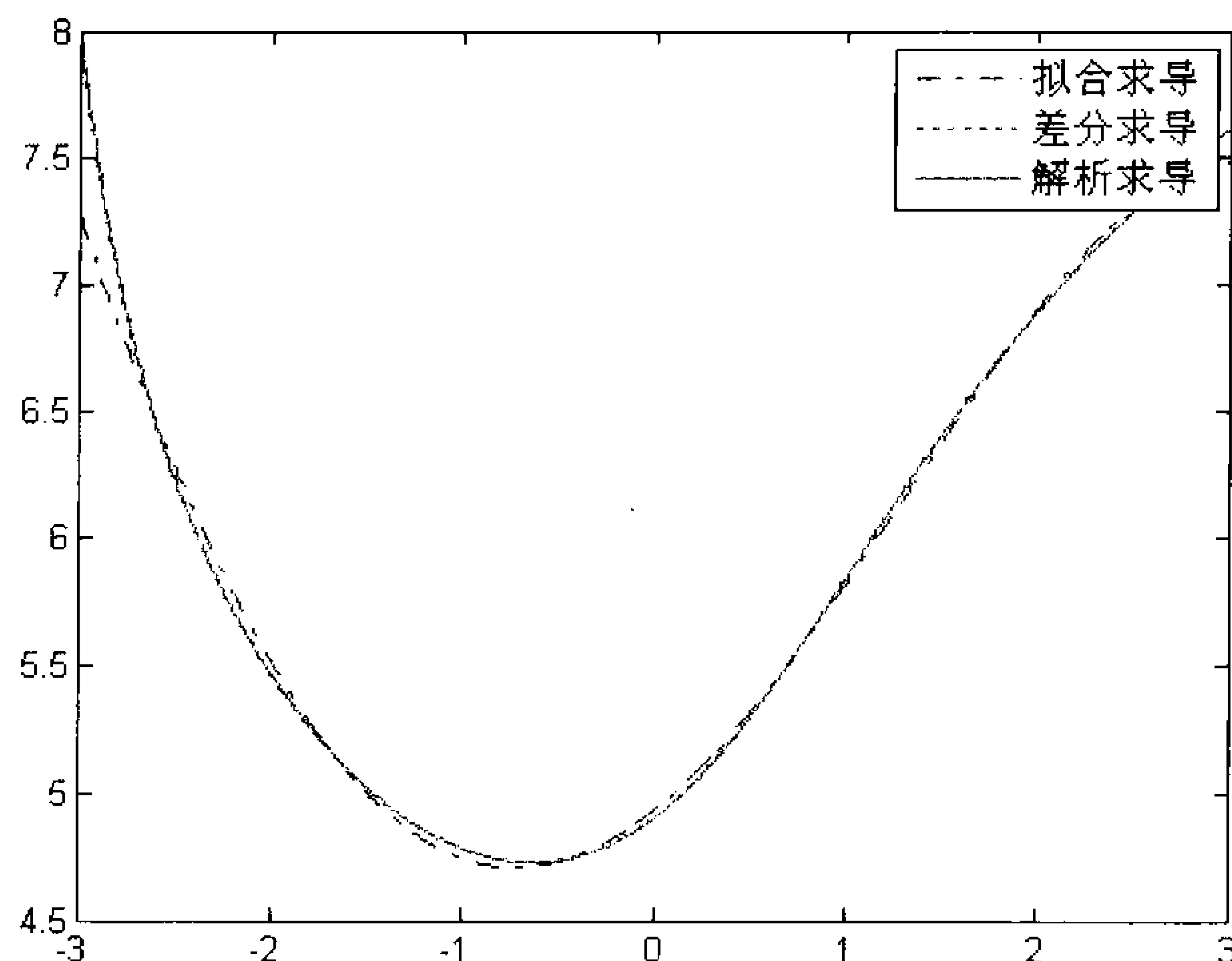


图 5-12 各种求导方法结果比较图

由图 5-12 可明显看出，差分求导方法的计算结果与解析结果非常接近，对离散数据先拟合后求导的计算方法求得的导数值与解析结果有较大误差。

5.3 求解线性方程组

在科学与工程计算中，大量的问题归结为求解线性代数方程组 $Ax=b$，其中 $A=\left(a_{ij}\right)\in R^{n\times n}$，$b=\left(b_1,b_2,\cdots,b_n\right)^T\in R^n$，$x=\left(x_1,x_2,\cdots,x_n\right)^T\in R^n$ 分别称为方程组的系数矩阵、右端向量和解向量。若 A 可逆，则方程组存在唯一解。当右端向量 b 为零向量时，该线性方程组称做齐次线性方程组，否则，称做非齐次线性方程组。

5.3.1 齐次线性方程组的求解

对于齐次线性方程组 AX=0 而言，可以通过求系数矩阵 A 的秩来判断解的情况。

（1）如果系数矩阵的秩等于 n（方程组中未知数的个数），则方程组只有零解。

（2）如果系数矩阵的秩小于 n，则方程组有无穷多解。

可以利用 MATLAB 函数 null(A)，也可以通过对系数矩阵进行行变换变成行最简形式来求得方程组 AX=0 的解。

例 5.13 求方程组 $\begin{cases}x_1-x_2+x_3+x_4=0\\x_1-x_2+x_3-2x_4=0\\x_1-x_2-2x_3+x_4=0\end{cases}$ 的解。

程序设计：

在 MATLAB 文本编辑窗口编制 M 文件：

```
clear
A=[1 -1 1 1;1 -1 1 -2;1 -1 -2 1];
```

```
format rat
n=4;
RA=rank(A)
if RA==n
    fprintf('方程组只有零解')
else
    B=null(A,'r')              %'r'表示解空间的有理基。
end
```

运行结果：

```
RA =
     3
B =
     1
     1
     0
     0
>> syms k                    %声明变量 k
>> X=k*B
X =
[ k]
[ k]
[ 0]
[ 0]
```

例 5.14 利用行变换求方程组 $\begin{cases}-x_1-2x_2+4x_3=0\\2x_1+x_2+x_3=0\\x_1+x_2-x_3=0\end{cases}$ 的解。

程序设计如下。

```
>> A=[-1 -2 4;2 1 1;1 1 -1];
>> rank(A)
ans =
     2
>> rref(A)
ans =
     1          0          2
     0          1         -3
     0          0          0
```

程序说明：

（1）由于 rank(A)=2，可以得出方程组有无穷多组解。

（2）由 A 的行最简形矩阵写出方程组的解为$[-2k\ 3k\ k]^T$。

5.3.2 非齐次线性方程组的求解

对于非齐次线性方程组 $AX=b$ 而言，则要根据系数矩阵 A 的秩和增广矩阵 $B=[A\ \ b]$的秩和未知数个数 n 的关系，才能判断方程组 $AX=b$ 的解的情况。

（1）如果系数矩阵的秩等于增广矩阵的秩等于 n，则方程组有唯一解。

（2）如果系数矩阵的秩等于增广矩阵的秩小于 n，则方程组有无穷多解。

（3）如果系数矩阵的秩小于增广矩阵的秩，则方程组无解。

对于非齐次线性方程组 $AX=b$ 而言，首先，应判断方程组的解的情况。其次，若有解，先求出方程组的特解。再次，求出对应齐次方程组的通解。最后，写出非齐次方程组的通解，即特解加对应齐次方程组的通解。

求 $AX=b$ 对应的齐次方程组 $AX=0$ 的通解，可以利用函数 null 或对系数矩阵 A 实行行变换；求 $AX=b$ 的特解，方法就比较多，根据方程组中方程的个数 m 和未知数的个数 n，可以把方程组求 $AX=b$ 分为：超定方程组（$m>n$）、恰定方程组（$m=n$）、欠定方程组（$m<n$）。我们可以根据系数矩阵 A 的性质选用适当的计算方法，如可以用“\”、系数矩阵 A 的逆或伪逆，或者利用 LU 分解、Cholesky 分解等。求解恰定方程组，可以使用“\”、inv（或 pinv）、LU 分解或者 Cholesky 分解。求解超定或欠定方程组，可以使用“\”或 pinv。下面主要通过例子加以介绍。

例 5.15 求方程组 $\begin{cases} 4x_1+2x_2-x_3=2 \\ 3x_1-x_2+2x_3=10 \\ 11x_1+3x_2=8 \end{cases}$ 的解。

程序设计：

在 MATLAB 文本编辑窗口编制 M 文件：

```
clear
A=[4 2 -1;3 -1 2;11 3 0];
b=[2 10 8]';
B=[A b];
n=3;
RA=rank(A)
RB=rank(B)
if RA==RB&RA==n
    X=A\b
    D=null(A,'r')
else
    fprintf('方程组无解')
end
```

运行结果：

```
RA =
     2
RB =
     3
方程组无解
```

程序说明：

（1）由于系数矩阵的秩为 2，矩阵是不满秩矩阵，其行列式 det(A)=0，方程组无解。

（2）求解系数矩阵 A 为方阵的非齐次线性方程组，可以利用“\”、inv（或 pinv）、LU 分解或者 Cholesky 分解，根据系数矩阵的性质选择不同的计算方法。

（3）在 MATLAB 中求解时，“\”所用的时间比 LU 分解或者 Cholesky 分解要长，但是比 inv（或 pinv）所用的时间要短。

例 5.16 利用 LU 分解求方程组$\begin{cases} x_1 + 2x_2 + 3x_3 = 1 \\ 2x_1 + x_2 + 2x_3 = 1 \\ x_1 + 3x_2 + 4x_3 = 1 \end{cases}$的解。

程序设计：

```
>> A=[1 2 3;2 1 2;1 3 4];
>> b=[1 1 1]';
>> B=[A b];
>> RA=rank(A)
RA =
      3
>> RB=rank(B)
RB =
      3
>> %RA==RB，可以判断出方程组有唯一解
>> [L,U]=lu(A)
L =
    0.5000    0.6000    1.0000
    1.0000         0         0
    0.5000    1.0000         0
U =
    2.0000    1.0000    2.0000
         0    2.5000    3.0000
         0         0    0.2000
>> X=U\(L\b)
X =
   -0.0000
   -1.0000
    1.0000
```

程序说明：

（1）利用 LU 分解比用“\”更节省时间。

（2）由 $RA=3$，可知矩阵 $\boldsymbol{A}$ 可逆，现用 $\boldsymbol{A}$ 的逆求解方程组：

```
>> A=[1 2 3;2 1 2;1 3 4];
>> B=inv(A);
>> b=[1 1 1]';
>> X=inv(A)*b
X =
   -0.0000
   -1.0000
    1.0000
```

例 5.17 求非齐次线性方程组$\begin{cases} 2x_1 + 3x_2 + x_3 = 4 \\ x_1 - 2x_2 + 4x_3 = -5 \\ 3x_1 + 8x_2 - 2x_3 = 13 \\ 4x_1 - x_2 + 9x_3 = -6 \end{cases}$的解。

程序设计：

在 MATLAB 文本编辑窗口编制 M 文件：

```
clear
A=[2 3 1;1 -2 4;3 8 -2;4 -1 9];
b=[4 -5 13 -6]';
B=[A b];
n=3;
RA=rank(A)
RB=rank(B)
if RA==RB&RA==n
    X=A\b
elseif RA==RB&RA<n
    C=A\b
    D=null(A,'r')
else
    fprintf('方程组无解')
end
```

运行结果：

```
RA =
     2
RB =
     2
Warning: Rank deficient, rank = 2    tol =      8.9702e-015.
C =
             0
        1.5000
       -0.5000
D =
    -2
     1
     1
>> syms k                          %声明变量 k
>> X=C+k*D                         %AX=b 的通解
X =
    [    -2*k]
    [   3/2+k]
    [ -1/2+k]
```

程序说明：

（1）在程序运行过程中，出现了警告信息，说明矩阵 A 是不满秩矩阵。

（2）系数矩阵 $\boldsymbol{A}$（$m \times n$）不是方阵，它的行数大于列数（$m>n$），且其秩 $RA<n$，如果方程组不存在精确解，所求得的特解 $C=A\backslash b$ 是方程组最小二乘意义上的解，可以通过本例进行验证：

```
>> A*C-b
ans =
  1.0e-014 *
   -0.3109
         0
   -0.7105
   -0.2665
```

由计算的结果可以看出，C 是方程组最小二乘意义上的解。

（3）由于系数矩阵 **A** 不是方阵，也可以用 pinv 函数求伪逆，所得结果是方程组最小二乘意义上的解。

```
>> C1=pinv(A)*b
C1 =
    0.3333
    1.3333
   -0.6667
>> A*C1-b
ans =
  1.0e-014 *
   -0.0888
   -0.0888
    0.1776
   -0.1776
```

由 **A***C1-*b* 的值可以得出，*C*1 不是方程组的精确解，但是它在工程中是具有实际意义的。

（4）如果系数矩阵 **A**（$m\times n$，$m>n$）是列满秩矩阵（$RA=n$），则下面 3 种方法计算特解：

x=**A**\b

x=pinv(**A**)**b*

x=inv(**A**'***A**)***A**'**b*

在 **A** 列满秩的情况下，3 种解法求出的解是相同的。

5.3.3 线性方程组的迭代计算

在计算机大规模集成电路设计、结构分析、网络理论、电力分布系统和图论，特别是数值求解多维偏微分方程组中，常常会遇到大规模的稀疏的线性代数方程组（其系数矩阵是非零元素占很小百分比的稀疏矩阵）。下面看一个形成大型方程组的例子。考虑 Poisson 方程 $u_{xx}+u_{yy}=0, 0\leqslant x\leqslant 1, 0\leqslant y\leqslant 1$，的离散逼近，其边界条件如下：

$$u(0,y)=y^2, u(1,y)=1, 0<y<1$$
$$u(x,0)=x^2, u(x,1)=1, 0<x<1$$

取 $\triangle x=\triangle y=0.25$ 进行网格划分，用二阶均差代替二阶导数，按逐行自左至右和自上而下的自然次序离散化可得下列线性方程组：

$$\begin{bmatrix} 4 & -1 & & -1 & & & & & \\ -1 & 4 & -1 & & -1 & & & & \\ & -1 & 4 & & & -1 & & & \\ -1 & & & 4 & -1 & & -1 & & \\ & -1 & & -1 & 4 & -1 & & -1 & \\ & & -1 & & -1 & 4 & & & -1 \\ & & & -1 & & & 4 & -1 & \\ & & & & -1 & & -1 & 4 & -1 \\ & & & & & -1 & & -1 & 4 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{bmatrix}=\begin{bmatrix} 0.125 \\ 0.25 \\ 1.5625 \\ 0.25 \\ 0 \\ 1 \\ 1.5625 \\ 1 \\ 2 \end{bmatrix}$$

其中，$ui+3j-3\left(i,j=1,2,3\right)$ 是 $u\left(i\Delta x, j\Delta y\right)$ 的近似值。这是一种特殊形状的稀疏矩阵。随着 Δx 和 Δy 的减小，所得到的方程组的阶数将增大。

对于大型线性代数方程组，常常用迭代方法进行计算。迭代法有存储空间小、程序简单等特点，在使用时，能保持系数矩阵的稀疏性不变。比较常用的迭代方法有 Gauss-Seidel 迭代法、Jacobi 迭代法和 SOR 方法。下面只介绍 Jacobi 迭代法。

已知线性方程组 $Ax=b$，记 $A=\left(a_{ij}\right)$，可以把 A 分解为：

$$\boldsymbol{A}=\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{U}$$

其中，$\boldsymbol{A}=\mathrm{diag}(a_{11},a_{22},\cdots,a_{nn})$，$\boldsymbol{L}=-\begin{bmatrix}0 & & & \\ a_{21} & 0 & & \\ \vdots & \ddots & \ddots & \\ a_{n1} & \cdots & a_{n,n-1} & 0\end{bmatrix}$，$\boldsymbol{U}=-\begin{bmatrix}0 & a_{12} & \cdots & a_{1n} \\ & 0 & \ddots & \vdots \\ & & \ddots & a_{n-1,n} \\ & & & 0\end{bmatrix}$

现设 D 非奇异，即 $a_{ii}\neq 0$，i=1，2，…，n。方程组 $\boldsymbol{A}x=b$ 等价于：

$$x=\boldsymbol{D}^{-1}\left(\boldsymbol{L}+\boldsymbol{U}\right)x+\boldsymbol{D}^{-1}b$$

由此构造迭代公式：

$$x^{(k+1)}=\boldsymbol{B}_{\mathrm{J}}x^{(k)}+f_{\mathrm{J}}，\ \mathrm{k}=0，1，\cdots，n \tag{5.7}$$

其中，迭代矩阵 $\boldsymbol{B}_{\mathrm{J}}$ 和向量 f_{J} 为：

$$\boldsymbol{B}_{\mathrm{J}}=\boldsymbol{D}^{-1}\left(\boldsymbol{L}+\boldsymbol{U}\right)=I-\boldsymbol{D}^{-1}\boldsymbol{A}$$

$$f_{\mathrm{J}}=\boldsymbol{D}^{-1}b$$

称式（5.7）为 Jacobi 迭代法，简称 J 法。

线性方程组的 Jacobi 迭代计算求解，可以通过编制下面的 MATLAB 函数方便地实现：

```
function x=jacobi_f(A,b,x0,tol,max)
%用 Jacobi 迭代法求解方程组 Ax=b
%输入：A 是一个 nxn 系数矩阵，b 是 nx1 的右端向量
%输出：x 是 nx1 的解向量
%迭代终止准则<tol，最大迭代次数 max
[n,m]=size(A);
xold=x0;C=-A;
for i=1:n
    C(i,i)=0;
end
for i=1:n
    C(i,:)=C(i,:)/A(i,i);
end
for i=1:n
    d(i,1)=b(i)/A(i,i);
end
```

```
    i=1;
    while i<=max
        xnew=C*xold+d;
        if norm(xnew-xold)<=tol
            x=xnew;
            disp('Jacobi 迭代法收敛')
            return;
        else
            xold=xnew;
        end
        disp([xnew']);
        i=i+1;
    end
    disp('Jacobi 迭代法不收敛');
    disp('最大迭代次数后的结果为')
    x=xnew;
```

例 5.18 用 Jacobi 方法求解下列方程组，设 $x(0)=0$，精度为 10^{-6}。

$$\begin{cases}10x_1 - x_2 = 9 \\ -x_1 + 10x_2 - 2x_3 = 7 \\ -2x_2 + 10x_3 = 6\end{cases}$$

解：在 MATLAB 指令窗口输入：

```
>> clear
>> a=[10 -1 0;-1 10 -2;0 -2 10];
>> b=[9;7;6];
>> tol=1e-6;
>> jacobi_f(a,b,[0;0;0],tol,100)
```

按回车键后，得到方程组的迭代计算结果：

```
        0.9000    0.7000    0.6000
        0.9700    0.9100    0.7400
        0.9910    0.9450    0.7820
        0.9945    0.9555    0.7890
        0.9956    0.9572    0.7911
        0.9957    0.9578    0.7914
        0.9958    0.9579    0.7916
        0.9958    0.9579    0.7916
        0.9958    0.9579    0.7916
        0.9958    0.9579    0.7916
    Jacobi 迭代法收敛
    ans =
        0.9958
        0.9579
        0.7916
```

由结果可知，共进行了 7 次迭代计算，结果在误差范围内收敛。如不需显示每一步迭代计算结果，可将 jacobi_f()函数中的语句“disp([xnew']);”不运行即可。

5.4 求解非线性方程和方程组

方程求根是初等数学的重要内容之一，也是科学和工程中经常碰到的数值计算问题。它的一般形式是求下列方程的根：

$$f(x)=0 \tag{5.8}$$

实际上，就是寻求使函数 $f(x)$ 取 0 的变量 x，所以求方程（5.8）的根，也叫求函数 $f(x)$ 的零点。如果变量 x 是列阵，方程（5.8）就代表方程组。

当方程（5.8）中的函数 $f(x)$ 是有限个指数、对数、三角、反三角或幂函数的组合时，则方程（5.8）被称为超越方程，例如，$\mathrm{e}^{-x}-\sin(\pi x/2)+\ln x=0$ 就是超越方程。

当方程（5.8）中的函数 $f(x)$ 是多项式时，即 $f(x)=P_n(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$，则方程（5.8）就成为下面的多项式方程，也称代数方程：

$$P_n(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0=0 \tag{5.9}$$

$P_n(x)$ 的最高次数 n 等于 2、3 时，用代数方法可以求出方程（5.9）的解析解，但是，当 $n\geqslant5$ 时，伽罗瓦（Galois）定理已经证明它是没有代数求根方法的。至于超越方程，通常很难求出其解析解。所以，方程（5.8）的求解经常使用作图法或数值法，而计算机的发展和普及为这些方法提供了一个广阔的发展前景，使之成为科学和工程中最实用的方法之一。

本节首先介绍求解 $f(x)=0$ 的 MATLAB 符号法求解指令，然后介绍求解 $f(x)=0$ 的 MATLAB 数值法求解指令。

5.4.1 求解 $f(x)=0$ 的 MATLAB 符号法

MATLAB 中设有求出方程 $f(x)=0$ 解析解或精确解的符号法指令 solve，由它得出的符号量结果，可以转换为任意位有效数字的数值解。该指令的使用格式如下：

solve(s1,s2,…sn,'v1','v2',…,'vn')

solve(s1,s2,…sn,'v1,v2,…,vn')

[z1,z2,…,zn]=solve(s1,s2,…sn, 'v1','v2',…,'vn')

①输入参量 s1,s2,…sn 为待解方程组 $f(x)=0$ 或函数 $f(x)$ 的字符、符号表达式，或者是代表它们的变量名。待解方程可以是任意线性、非线性或超越方程。

②输入参量 v1,v2,…,vn 是与方程对应的未知量，它的数目必须与方程数目相等；若写有输出变量名 z1,z2,…,zn 且与方程数相等，则输入变量 v1,v2,…,vn 可以默认。

③输出参量 z1,z2,…,zn 是指定的输出变量名，方程解的结果分别赋值给它们。但是赋值顺序并不是输入变量 v1,v2,…,vn 的排序，而是按未知变量名在字母表中的排序输出。求解方程组时，这些输出参量不可省略，而且必须跟方程数相等，否则只输出解的结构（解的维数）。

④当方程组不存在解析解或精确解时，该命令输出方程的数字形式符号量解。

⑤解析表达式太冗长或含有不熟悉的特殊函数时，可用 vpa 指令转换成数值解。

例 5.19 由方程 $ax^2+bx+5=0$ 求出 x 和 b。

程序设计：

```
>> s1='a*x^2+b*x+5';                %函数 f(x)的字符串表达式
或 s1='a*x^2+b*x+5=0';              %方程 f(x)=0 的字符串表达式
或 s1=sym('a*x^2+b*x+5');           %函数 f(x)的符号表达式
或 s1=sym('a*x^2+b*x+5=0');         %方程 f(x)=0 的符号表达式
>> x=solve(s1)
 x =
[ 1/2/a*(-b+(b^2-20*a)^(1/2))]
[ 1/2/a*(-b-(b^2-20*a)^(1/2))]
>> b=solve(s1,'b')
 b =
 -(a*x^2+5)/x
```

例 5.20 求解方程组 $\begin{cases} x^2+x\sqrt{5}=-1 \\ x+3z^2=4 \\ yz+1=0 \end{cases}$。

程序设计：

```
>> s1='x^2+x*sqrt(5)=-1';s2='x+3*z^2=4';s3='y*z+1=0';
>> [u v w]=solve(s1,s2,s3)
 u =
[   1/2-1/2*5^(1/2)]
[   1/2-1/2*5^(1/2)]
[ -1/2-1/2*5^(1/2)]
[ -1/2-1/2*5^(1/2)]
v =
[   1/44*(42+6*5^(1/2))^(1/2)*(-7+5^(1/2))]
[ -1/44*(42+6*5^(1/2))^(1/2)*(-7+5^(1/2))]
[   1/76*(54+6*5^(1/2))^(1/2)*(-9+5^(1/2))]
[ -1/76*(54+6*5^(1/2))^(1/2)*(-9+5^(1/2))]
 w =
[   1/6*(42+6*5^(1/2))^(1/2)]
[ -1/6*(42+6*5^(1/2))^(1/2)]
[   1/6*(54+6*5^(1/2))^(1/2)]
[ -1/6*(54+6*5^(1/2))^(1/2)]
>> u=vpa(u,5),v=vpa(v,5),w=vpa(w,5)
u =
[ -.61810]
[ -.61810]
[ -1.6181]
[ -1.6181]
v =
[ -.80599]
[   .80599]
[ -.73076]
[   .73076]
w =
[   1.2407]
```

```
[ -1.2407]
[  1.3685]
[ -1.3685]
```

5.4.2 方程 $f(x)=0$ 数值解的 MATLAB 实现

MATLAB 中求方程数值解的方法很多，有的是专用指令，有的是根据方程性质而借用其他专用指令求得的，使用中应特别注意应用条件及它们之间的差异。

1．代数方程的求根指令 roots

对于多项式方程（5.9），可用多项式求根指令 roots 求解，使用格式如下：

```
roots(p)
```

①每次只能求一个一元多项式的根，该指令不能用于求方程组的解，必须把多项式方程变成 $P_n(x)=0$ 的形式。

②参数 p 是多项式 $P_n(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$ 的系数向量 $p=[a_n,a_{n-1},\cdots a_1,a_0]$，该向量的分量由多项式系数构成，排序是从高次幂系数到低次幂系数，缺少的幂次系数用 0 填补。

③输出多项式方程的所有实数和复数根。

例 5.21 求方程 $x^3=x^2+1$ 的解。

程序设计：

```
>> p=[1 -1 0 -1];
>> roots(p)
ans =
    1.4656
      -0.2328 + 0.7926i
      -0.2328 - 0.7926i
```

2．求函数零点指令 fzero

求解方程 $f(x)=0$ 的实数根也就是求函数 $f(x)$ 的零点。MATLAB 中设有求函数 $f(x)$ 零点的指令 fzero，可用它来求方程的实数根。该指令的使用格式如下：

```
fzero(fun,x0,options)
```

①输入参数 fun 为函数 $f(x)$ 的字符表达式、内联函数名或 M-函数文件名。

②输入参数 x0 为函数某个零点的大概位置（不要取 0）或存在的区间 $[x_i,x_j]$，要求函数 $f(x)$ 在 x0 点左右变号，即 $f(x_i)f(x_j)<0$。

③输入参数 options 可有多种选择。options 的设置如下：

options=optimset('param1',value1,'param2',value2,...)

式中 param 共有 29 项，详见“help optimset”。当 options 如下进行设置时：

options=optimset('disp','iter')

将输出寻找零点的中间数据。

④该指令无论是对多项式函数还是超越函数都可以使用，但是每次只能求出函数的一个零点，因此在使用前应该摸清函数零点的个数和存在的大体范围。为此，常用绘图指令 plot、fplot 或 ezplot 画出函数 $f(x)$ 的曲线，从图上估计出函数零点的位置。

例 5.22 求方程 $x^2+4\sin(x)=25$ 的实数根 $(-2\pi < x < 2\pi)$。

解：（1）首先确定方程实数根存在的大致范围。为此，先将方程变成标准形式 $f(x)=x^2+4\sin(x)-25$。在指令窗口中输入：

```
>> p=[1 -1 0 -1];
>> clf,ezplot x-x,grid,hold,ezplot('x^2+4*sin(x)-25')
Current plot held
```

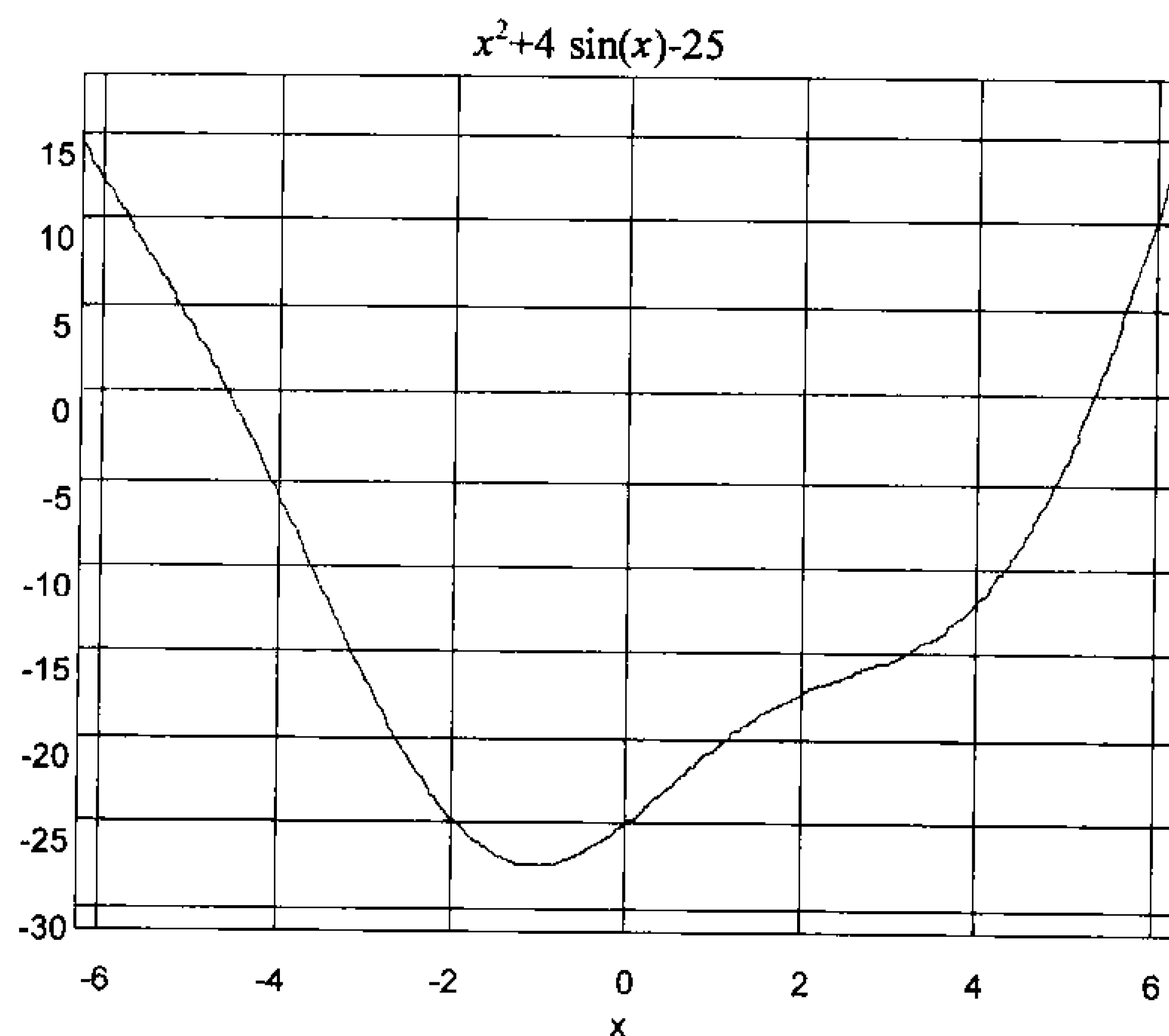


图 5-13 确定方程实数根存在位置的曲线

从曲线上可以看出，函数的零点大约在 $x_1\approx -4$ 和 $x_2\approx 5$ 附近。

（2）直接使用指令 fzero 求出方程在 $x_1\approx -4$ 时的根。在指令窗口中输入：

```
>> x1=fzero('x^2+4*sin(x)-25',-4)
x1 =
   -4.5861
```

若输入：

```
>> options=optimset('disp','iter');
>> fzero('x^2+4*sin(x)-25',-4,options)
 Func-count      x          f(x)          Procedure
    1            -4       -5.97279         initial
    2      -3.88686       -7.17961         search
    3      -4.11314       -4.77907         search
    4         -3.84       -7.68241         search
    5         -4.16       -4.28931         search
    6      -3.77373       -8.39553         search
    7      -4.22627       -3.60199         search
    8         -3.68       -9.40652         search
    9         -4.32       -2.64161         search
   10      -3.54745       -10.8364         search
   11      -4.45255       -1.30909         search
   12         -3.36       -12.8437         search
   13         -4.64       0.519124         search
 Looking for a zero in the interval [-3.36, -4.64]
   14      -4.59027      0.0408288         interpolation
   15      -4.58604 -9.62876e-005         interpolation
```

```
    16          -4.58605   4.13699e-008         interpolation
    17          -4.58605   4.61853e-014         interpolation
    18          -4.58605              0         interpolation
Zero found in the interval: [-3.36, -4.64].
ans =
   -4.5861
```

中间数据表明，求根过程中不断缩小探测范围，最后得出–4 附近满足精度的近似根。

（3）为了求出 $x_2 \approx 5$ 附近的根，在指令窗口中输入：

```
>> x2=fzero('x^2+4*sin(x)-25',5)
x2 =
    5.3186
```

（4）也可以用内联函数作为输入参数，如输入：

```
>> f=inline('x^2+4*sin(x)-25');x1=fzero(f,-4),x2=fzero(f,5)
x1 =
   -4.5861
x2 =
    5.3186
```

例 5.23 将方程 $x^2+4\sin(x)=25$ 编成 M-函数文件（实用中在函数较为复杂、而又多次重复调用时，才这样做），用 fzero 求解。

解：①在 MATLAB 文本编辑窗口编制 M 文件：

```
function yy=li4_6(x)
yy=x^2+4*sin(x)-25;
```

以“li4_6”为名存盘，退出编辑调试窗口，回到指令窗。

②在指令窗中输入作图指令：

```
>> fplot('li4_6',[-6,6]),grid,hold,ezplot x-x
```

按回车键。在图形窗口中得出与前面类似的函数曲线，从中可以确定根的大体位置。

③在指令窗口中输入：

```
>> x1=fzero('li4_6',-4),x2=fzero('li4_6',5)
x1 =
   -4.5861
x2 =
    5.3186
```

5.4.3 求解非线性方程组的数值解

fsolve 是用最小二乘法求解非线性方程组 $F(X)=0$ 的指令，变量 X 可以是向量或矩阵，方程组可以由代数方程或超越方程构成。它的使用格式如下：

```
fsolve('fun',X0,OPTIONS)
```

①参数 fun 是编辑并存盘的 M-函数文件名称，可以用@代替单引号对它进行标识。M-函数文件主要内容是方程组 $F(X)=0$ 中的函数 $F(X)$，即方程左边的函数。

②参数 X0 是向量或矩阵，为探索方程组解的起始点。求解将从 X0 出发，逐渐趋向，最终得到满足精度要求，最接近 X0 的近似根 $X^*:F(X^*)\approx 0$。由于 X0 是向量或矩阵，无法用画图方法进行估计，实际问题中常常是根据专业知识、物理意义等进行估计。

③该指令输出一个与 X0 同维的向量或矩阵，为方程组的近似数值解。

④参数 OPTIONS 为设置选项，用它可以设置过程显示与否、误差和算法等，具体内容可用 help 查阅。通常可以省略该项内容。

此外，还有其他一些调用形式，可用 help 查阅，如下：

[X,FVAL,EXITFLAG]=FSOLVE(FUN,X0,...);

[X,FVAL,EXITFLAG,OUTPUT]=FSOLVE(FUN,X0,...);

[X,FVAL,EXITFLAG,OUTPUT,JACOB]=FSOLVE(FUN,X0,...)。

例 5.24 求方程组 $\begin{cases} 3x=\cos(yz)+0.5 \\ 2x^2-81(y+0.1)^2+\sin z+1.06=0 \\ e^{-xy}+20z+\dfrac{10}{3}\pi=1 \end{cases}$，在 $x_0=0.1$，$y_0=0.1$，$z_0=-0.1$ 附近的数值解。

解：首先将方程组变换成 $F(X)=0$ 的形式，x，y，z 看做向量 X 的 3 个分量。

①在 MATLAB 文本编辑窗口编制 M 文件：

```
function output=li5_06261(X)
output(1)=3*X(1)-cos(X(2)*X(3))-0.5;
output(2)=2*X(1)^2-81*(X(2)+0.1)^2+sin(X(3))+1.06;
output(3)=exp(-X(1)*X(2))+20*X(3)+10/3*pi-1;
```

用“li5_06261”为 M-函数文件名存盘，退出编辑调试窗，回到指令窗。

②在指令窗中输入：

```
>> fsolve(@li5_06261,[0.1 0.1 -0.1],optimset('fsolve'))
Optimization terminated successfully:
 First-order optimality is less than options.TolFun.
ans =
      0.5000      0.0144      -0.5232
```

这是方程组的最小二乘解，用符号指令 solve 无法得出最终结果。

5.5 方阵特征值和特征向量的计算

在研究振动与波及自动控制中的稳定性等问题时，数学化后的求解常常归结为求一些矩阵的特征值和特征向量。矩阵特征值的计算方法分两类，一类是解多项式方程，即由矩阵得出其特征多项式和特征方程，特征方程的根就是矩阵的特征值。但是，当矩阵的阶数很高时，由于高次多项式方程的求根较为繁杂，而且重根将使计算精度降低，于是这一方法并不理想。另一类方法是迭代方法，它是构造一个极限为特征值和特征向量的收敛序列，序列中的每个元素都是特征值和不同误差特征向量的近似值。这种方法的迭代循环，给计算机求解带来了方便，所以广为流行。迭代计算方法又有好多种，基本原理雷同。比较常用的是雅克比法和 QR（正交－三角形分解）算法。

以矩阵为运算单元的 MATLAB 中，有许多求算矩阵特征参数的指令，使用非常方便。这里只介绍跟求矩阵特征值有关的几个常用指令及 QR 分解指令，要用其他指令时，可用 help 调出“矩阵代数（matfun）库”查找。

5.5.1 求矩阵特征值的有关指令

1．求方阵行列式的指令 det

方阵阶数很高时，手工演算求行列式值非常麻烦，MATLAB 中设有专用指令 det，它的调用格式如下：

```
det(A)
```

输入参量 A 必须是数值方阵，返回 A 的行列式$|A|$。

2．求方阵特征多项式的指令 poly

方阵 A 的特征多项式 det(A-λE) 是关于 λ 的代数多项式，它的根就是方阵的特征值。求方阵特征多项式的专用指令是 poly，调用格式如下：

```
p=poly(A)
```

①输入参量 A 必须是方阵。

②输出参量 p 是 A 的特征多项式系数向量。

③用 roots(p)可以求解矩阵 A 的特征值。

3．求方阵特征值和特征向量指令 eig

方阵特征值的求法有多种，上面介绍的先求特征多项式再求特征多项式的根就是其中之一。但是，在 MATLAB 中还有一个方便而专用的指令 eig，它同时可以得出特征值和一组特征向量。调用格式如下：

```
[x  r]=eig(A)
[x  r]=eig(A,"nobalance")
eig(A)
```

①输入参量 A 必须是方阵。

②输出参量 x 是一个矩阵，它的各列是方阵 A 的特征向量。

③输出参量 r 是一个对角阵，其元素是方阵 A 的特征值，r 与 x 同列向量相对应。

④当不写输出格式[x r]时，只输出由 A 的特征值为元素的列阵。

⑤如果 A 中含有小到跟截断误差相当的元素时，加写输入参数“nobalance”，它可以提高小元素的作用，通常可以省略该参数，以免使结果误差变大。

例 5.25 求方阵$a3=\begin{bmatrix}-2 & 1 & 1\\ 0 & 2 & 0\\ -4 & 1 & 3\end{bmatrix}$的行列式值、特征多项式、特征值和特征向量。

解：在 MATLAB 命令窗口输入：

```
>> a3=[-2 1 1;0 2 0;-4 1 3];det(a3)
```

按回车键得出：

```
ans =
      -4
```

$a3$ 的行列式$|a3|=-4$。

输入：

```
>> p=poly(a3)
```

按回车键得出：

```
p =
    1    -3     0     4
```

p 为 $a3$ 的特征多项式的系数向量，用 poly2str 指令可得出多项式表达式，输入：

```
>> p1=poly2str(p,'y')
```

按回车键得出：

```
p1 =
    y^3 - 3 y^2 + 4
```

求特征多项式的根，即解方程 $y^3 - 3y^2 + 4 = 0$，输入：

```
>> roots(p)
```

按回车键得出：

```
ans =
    2.0000
    2.0000
   -1.0000
```

$a3$ 的特征值是 2，2，−1。

不书写输出变量时，eig 指令只输出特征值，如输入：

```
>> eig(a3)
```

按回车键得出：

```
ans =
    -1
     2
     2
```

这是特征值排成的列阵。如果输入输出格式，一次就可同时得出特征值和特征向量。输入：

```
>> [x r]=eig(a3)
```

按回车键得出：

```
x =
   -0.7071   -0.2425    0.3015
         0         0    0.9045
   -0.7071   -0.9701    0.3015
r =
    -1     0     0
     0     2     0
     0     0     2
```

也可以先把 $a3$ 变成符号矩阵，再用 eig 求解，这样可以得出整数特征值。

输入：

```
>> [x r]=eig(sym(a3))
```

按回车键得出：

```
x =
    [ 1,  0,  1]
    [ 4, -1,  0]
    [ 0,  1,  1]
r =
    [ 2,  0,  0]
    [ 0,  2,  0]
```

```
[ 0, 0, -1]
```

5.5.2 QR 算法与矩阵的正交分解指令 qr

由线性代数中的施密特（Schmidt）正交化方法可以推得，任意 n 阶方阵 A 可以分解为一个正交矩阵 $\boldsymbol{Q}(\boldsymbol{Q}^T\boldsymbol{Q}=\boldsymbol{E})$ 和一个上三角阵 R 的乘积：

$$A=QR$$

这种把矩阵分解为正交阵和上三角阵乘积的过程，叫做正交三角分解或 QR 分解。如果 $\boldsymbol{A}$ 是非奇异方阵，则这种分解是唯一的。QR 算法的理论基础就是矩阵的正交三角分解。

若 $\boldsymbol{A}$ 是一个方阵，设 $A_1=A$，A_1 可以分解为正交阵 Q_1 和上三角阵 R_1 之积：

$$A_1=Q_1R_1$$

将 Q_1、R_1 顺序颠倒，令 $A_2=R_1Q_1=Q_1^{-1}A_1Q_1$，再对 A_2 重复上述步骤，得到 A_3,…不断重复这种变换，则要 QR 算法的计算公式：

$$\begin{cases}\boldsymbol{A}_k=\boldsymbol{Q}_k\boldsymbol{R}_k,\ \ \boldsymbol{Q}_k^T\boldsymbol{Q}_k=\boldsymbol{E} \\ \boldsymbol{A}_{k+1}=\boldsymbol{R}_k\boldsymbol{Q}_k=\boldsymbol{Q}_k^{-1}\boldsymbol{A}_k\boldsymbol{Q}_k\end{cases} k=1,2,\cdots$$

E 为 n 阶单位方阵。由此可以得出方阵序列 $\boldsymbol{A}_1$，$\boldsymbol{A}_2,\cdots$，$\boldsymbol{A}_k,\cdots$。这个序列的每个方阵都与方阵 A 相似。理论证明，当 $k\to\infty$ 时，在一定的条件下方阵序列 $\boldsymbol{A}_1$，$\boldsymbol{A}_2,\cdots$，$\boldsymbol{A}_k,\cdots$ 的主对角线元素趋于方阵 A 的特征值。

QR 算法的收敛速度是线性的，而且运算量很大。但是它不限定方阵 A 必须对称，有一定实用价值；当然，实际应用中还需进行许多改进，这里不再介绍。

在 MATLAB 中有用于矩阵 QR 分解的专用指令 qr。用 qr 指令可以实现矩阵 a 的正交三角分解，调用格式如下：

[q r]=qr(a)

[q r p]=qr(a)

①输入参数矩阵 a 不必是方阵。

②输出参量用[q r]格式（可以用其他字母）时，q 为正交方阵，阶数等于 a 的行数和列数中的较小者，满足 $q^Tq=E$；r 为与 a 同维的上三角阵，满足 ***qr**=a*。

③输出参量用[q r p]格式时，q 为正交方阵，r 为对角线元素绝对值递减的上三角阵，p 为换位阵，使之满足 a×p=q×r。

如果不写输出格式时，将输出一个变换过的矩阵，在此不予介绍。

例 5.26 求矩阵 $A2=\begin{bmatrix}1&2&4&1&3\\5&1&2&7&4\\6&2&1&4&8\end{bmatrix}$ 的正交三角分解。

解：在 MATLAB 命令窗口输入：

```
>> A2=[1 2 4 1 3;5 1 2 7 4;6 2 1 4 8];
>> [q r]=qr(A2)
```

按回车键得出：

```
q =
   -0.1270    0.9501   -0.2850
   -0.6350   -0.2986   -0.7125
   -0.7620    0.0905    0.6412
r =
   -7.8740   -2.4130   -2.5400   -7.6200   -9.0170
         0    1.7825    3.2936   -0.7782    2.3797
         0         0   -1.9237   -2.7074    1.4249
```

为了验证 q 的正交性，可输入：

```
>> q*q'
```

按回车键得出：

```
ans =
    1.0000   -0.0000   -0.0000
   -0.0000    1.0000   -0.0000
   -0.0000   -0.0000    1.0000
```

如果输入：

```
>> [q1 r1 p]=qr(A2)
```

按回车键得出：

```
q1 =
   -0.3180    0.2429   -0.9165
   -0.4240   -0.9010   -0.0916
   -0.8480    0.3594    0.3895
r1 =
   -9.4340   -6.6780   -2.9680   -2.7560   -7.5260
         0   -4.6265   -0.4712    0.3036   -2.1056
         0         0   -3.4596   -1.1456    0.9623
p =
     0     0     0     0     1
     0     0     0     1     0
     0     0     1     0     0
     0     1     0     0     0
     1     0     0     0     0
```

如果输入：

```
>> q1*r1
```

按回车键得出：

```
ans =
    3.0000    1.0000    4.0000    2.0000    1.0000
    4.0000    7.0000    2.0000    1.0000    5.0000
    8.0000    4.0000    1.0000    2.0000    6.0000
```

再输入：

```
>> A2*p
```

按回车键得出：

```
ans =
     3     1     4     2     1
     4     7     2     1     5
     8     4     1     2     6
```

可见，A2*p 与 A2 只是元素位置不同而已，p 起到了对 A2 元素位置的调换作用，所

以称换位矩阵。结果满足 A2*p=q1*r1。

5.6 常微分方程的求解

自然科学和工程技术的许多领域里，常会碰到常微分方程的定解问题。通常把含有自变量 x、未知的一元函数 $y(x)$及其导数或微分的方程，叫做常微分方程，一般形式是：

$$G(x,y,y',\cdots,y^{(m)})=0 \text{ 或 } y^{(m)}=f(x,y,y',\cdots,y^{(m-1)}) \tag{5.10}$$

方程中出现的函数最高阶导数 $y^{(m)}$ 的阶数 m，称为常微分方程的阶数。

求解常微分方程就是寻求一个解函数 $y=f(x)$，它能满足微分方程（5.10）。m 阶常微分方程的通解含有 m 个待定常数，如果给出了 m 阶常微分方程的 m 个初始条件：

$$y(x_0)=y_0, y'(x_0)=y_1,\cdots,y^{(m-1)}(x_0)=y_{m-1} \tag{5.11}$$

就可确定出这 m 个待定常数。式（5.11）中 $y_j=y^{(j)}(x_0)=\frac{\mathrm{d}^j y(x)}{\mathrm{d}x^j}\Big|_{x=x_0}(j=0,1,2,\cdots,m-1)$，$y^{(j)}$上标($j$)中的 j 表示导数阶数。

不含待定常数的解函数，称为常微分方程的特解。

微分方程的解函数 $y=f(x)$如果是解析表达式，称为微分方程的解析解；如果用表格法或图示法表示出函数关系 $y_i=F(x_i)(i=0,1,\cdots, n)$，满足或近似满足微分方程（5.10）和初始条件（5.11），就称它为微分方程初值问题的数值解。

除了少数几种类型的常微分方程可以求出解析解外，多数情况下都只能借助于数值解法得到近似解。有些常微分方程看似非常简单，例如，具有初始条件的一阶常微分方程：

$$\begin{cases}\dfrac{\mathrm{d}y(x)}{\mathrm{d}x}+2xy(x)=1\\ y(0)=0\end{cases} \tag{5.12}$$

很容易得到它的解为 $y(x)=\mathrm{e}^{-x^2}\int_0^x \mathrm{e}^{t^2}\mathrm{d}t$。但是，要得出它的最终解，还得进行数值积分。即便解法最简单的线性常系数微分方程，特征方程若是高次代数方程，求根也并非易事，有时还会把求解搞得更为复杂，更不必说是非线性微分方程了。因此，学习和掌握微分方程的数值解法，是非常必要且很有实用价值的。

5.6.1 求解常微分方程的 MATLAB 符号法

用 MATLAB 语言求解常微分方程的解析解有专用指令，用该指令可以求出常微分方程的通解和特解，非常方便，只是需要先把微分方程转化为符号法要求的标准形式。

1．常微分方程的 MATLAB 符号表示法

MATLAB 符号法要求常微分方程作如下形式上的变换。

（1）用“D*my*”表示函数 $y=f(x)$的 m 阶导数 $y^{(m)}=f^{(m)}(x)$。例如，D*y* 表示 y 对自变量的一阶导数 $\frac{\mathrm{d}y}{\mathrm{d}x}$ 或 $\frac{\mathrm{d}y}{\mathrm{d}t}$；D*my* 表示 y 对自变量的 m 阶导数 $\frac{\mathrm{d}^m y}{\mathrm{d}x^m}$ 或 $\frac{\mathrm{d}^m y}{\mathrm{d}t^m}$，式中的 D 必须得大写。据此，常微分方程（5.10）可以写成：

$$\mathrm{D}my=F(x,y,\mathrm{D}y,\mathrm{D}2y,\cdots,\mathrm{D}(m-1)y)$$

（2）初始条件的写法与上述的规定完全一致。因此式（5.11）可以写成：

$$y(x_0)=y_0,\mathrm{D}y(x_0)=y_1,\cdots,\mathrm{D}(m\text{-}1)y(x_0)=y_{m\text{-}1}$$

（3）不特别界定时，通常默认小写字母“*t*”为函数的自变量。

据上述规定，符号法可以把具有初始条件的一阶微分方程（5.12）表示成：

$$\mathrm{D}y+2xy=1,\ \ y(0)=0$$

2．求解常微分方程的符号法指令 dsolve

该指令的使用格式如下：

[y1,y2,⋯,y12]=dsolve(a1,a2,⋯,a12)

①每个输入参数 a1，a2，⋯，a12 都包含三部分内容：符号化的微分方程、符号化的初始条件和界定的自变量，每个部分都用单引号界定，两部分之间用逗号分隔，第一部分不得默认。

②当“初始条件”全部默认或部分默认时，输出带有待定常数的微分方程通解，待定常数的数目等于默认的初始条件数。待定常数用 C1，C2，⋯表示。

③当“界定的自变量”默认时，默认的自变量是小写“*t*”。

④由于每个输入参量 ai(*i*=1，2，⋯，12)中第一部分不限定于一个微分方程，参量 ai 又可以多达 12 个，所以该指令可以用于求解常微分方程组。

⑤输出参量只有在求解一个常微分方程时才可以默认，求解常微分方程组时不得默认，因为这时要输出多个函数，默认将无法区分。

例 5.27 求二阶微分方程 $\frac{\mathrm{d}^2 y}{\mathrm{d}x^2}+y=1-\frac{t^2}{\pi}$ 的通解及满足 $y(0)=0.2$，$y'(0)=0.5$ 的特解。

解：①将微分方程符号化，写成 D2y+y=1-t^2/pi。

②在 MATLAB 命令窗口输入：

```
>> y=dsolve('D2y+y=1-t^2/pi')
```

按回车键得出：

```
y =
 sin(t)*C2+cos(t)*C1+(2+pi-t^2)/pi
```

这是二阶微分方程的通解，即 $y=(\pi+2-t^2)\frac{1}{\pi}+\mathrm{C}_1\cos(t)+\mathrm{C}_2\sin(t)$，含有两个待定常数 C_1 和 C_2。由于方程中含有的自变量 *t* 正是默认的自变量“*t*”，所以省去了“界定的自变量”。

③求特解，在 MATLAB 命令窗口输入：

```
>> y=dsolve('D2y+y=1-t^2/pi','y(0)=0.2,Dy(0)=0.5')
```

按回车键得出：

```
y =
```

```
1/2*sin(t)-2/5*cos(t)*(2*pi+5)/pi+(2+pi-t^2)/pi
```

这是二阶微分方程的一个特解：

$$y=(\pi+2-t^2)\frac{1}{\pi}-\frac{4\pi+10}{5\pi}\cos(t)+\frac{\sin(t)}{2}$$

利用 MATLAB 画图指令，可以把它画成曲线，为此在命令窗口中输入：

```
>> ezplot('1/2*sin(t)-2/5*cos(t)*(2*pi+5)/pi+(2+pi-t^2)/pi',[-3 3]),grid
```

按回车键得出如图 5-14 所示曲线。

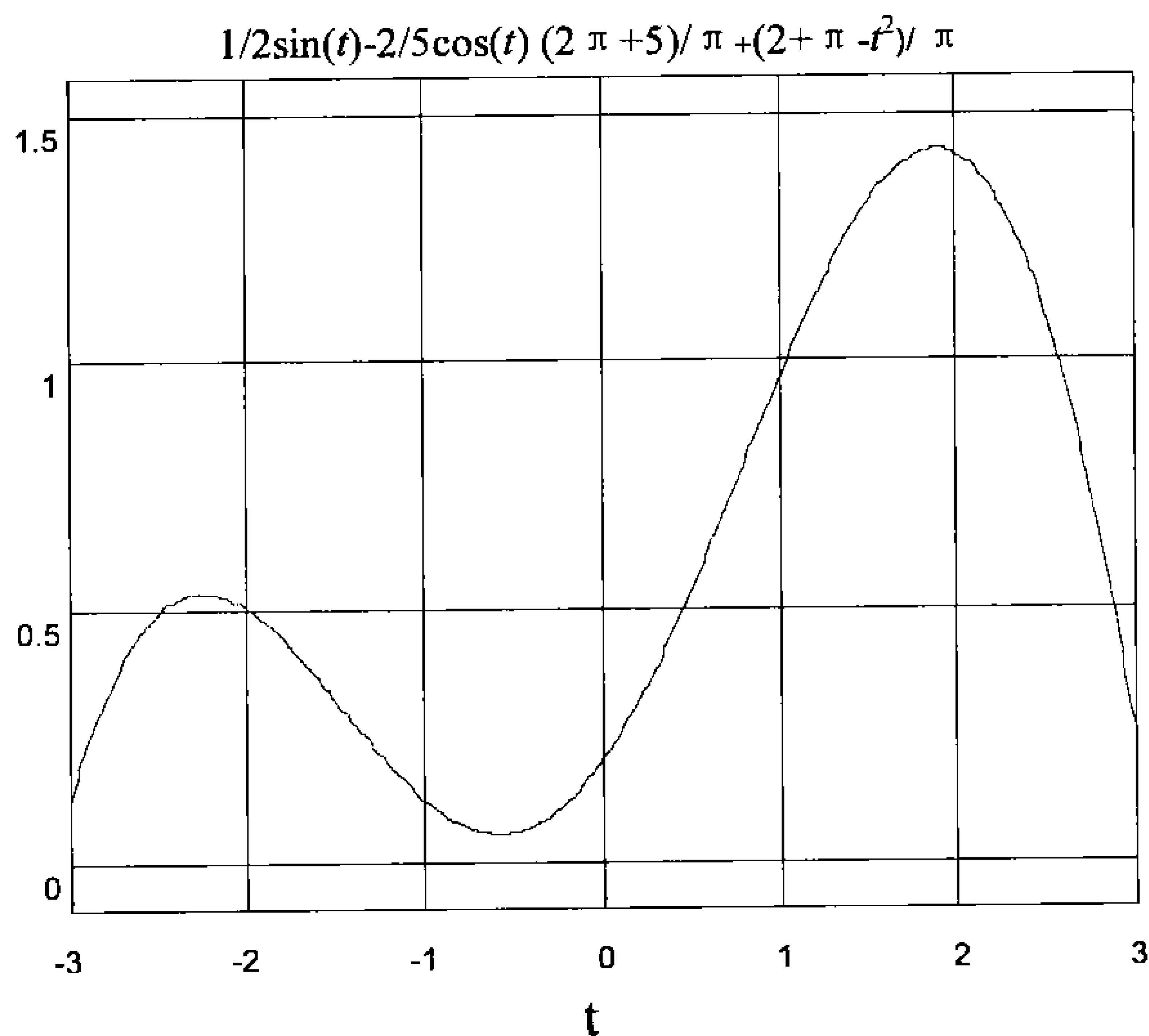


图 5-14　特解函数曲线

微分方程的通解是一组函数曲线，而特解是一条函数曲线，这条曲线上的任何一点对应的坐标值 t、函数值 $y=f(t)$ 和函数在该坐标点的导数值 $y'(t)$ 满足微分方程。当然，这条曲线也满足初始条件，如图 5-14 中，曲线的起点为 $y(0)=0.2$，$y'(0)=0.5$ 。

例 5.28　求 $\begin{cases}\dfrac{\mathrm{d}u}{\mathrm{d}s}=3u-2v\\ \dfrac{\mathrm{d}v}{\mathrm{d}s}+v=2u\end{cases}$ 的通解及满足初始条件 $u(0)=1$，$v(0)=0$ 的特解。

解：求通解时在 MATLAB 命令窗口输入：

```
>> [u v]=dsolve('Du=3*u-2*v,Dv=2*u-v')
```

按回车键得出：

```
u =
exp(t)*(C1+C2*t)
v =
1/2*exp(t)*(2*C1+2*C2*t-C2)
```

这是方程组的通解。未输入界定的自变量，输出结果则自动用了小写“t”。

求特解时，在输入参数中加进“初始条件”，输入：

```
>> [u v]=dsolve('Du=3*u-2*v,Dv=2*u-v','u(0)=1,v(0)=0','s')
```

按回车键得出：

```
u =
```

```
exp(s)*(1+2*s)
v =
2*exp(s)*s
```

特解的代数表达式分别为：$u=(1+2s)e^s$，$v=2se^s$。由于界定的自变量为“s”，所以特解的函数中以“s”作为自变量。

5.6.2 常微分方程的数值解

实际遇到的常微分方程，多数是很难找到解析解的。因此，必须学会用数值解法求出常微分方程的特解，即找出用表格或图示表示的解函数，近似满足微分方程和初始条件。

1．求常微分方程数值解的基本原理

一般的高阶微分方程总可以化为一阶微分方程组来求解。例如，m 阶微分方程初值问题：

$$\begin{cases} y^{(m)}=f(x,y,y',\cdots,y^{(m-1)}) \\ y(x_0)=y_0,y'(x_0)=y_1,\cdots,y^{(m-1)}(x_0)=y_{m-1} \end{cases}$$

只要引入新变量 $y_1=y,y_2=y',\cdots,y_m=y^{(m-1)}$，就可以变换成一阶微分方程组：

$$\begin{cases} y_1'=y_2 \\ y_2'=y_3 \\ \quad\vdots \\ y_{m-1}'=y_m \\ y_m'=f(x,y_1,y_2,\cdots,y_m) \end{cases}$$

初始条件为：

$$y_1(x_0)=y(x_0)=y_0,y_2(x_0)=y'(x_0)=y_1,\cdots,y_m(x_0)=y^{(m-1)}(x_0)=y_{m-1}$$

由于高阶微分方程可以变换为由多个一阶微分方程组成的微分方程组，所以只讨论一阶常微分方程初值问题的数值解。

求一阶常微分方程的数值解，就是找出用表格法或图示法表示的解函数 $y_i=y(x_i)$。为此先把自变量离散化成 $x_1,x_2,\cdots$，找出与其对应的函数值 $y_1,y_2,\cdots$，如表 5-1 所示。

表 5-1　函数 $y=f(x)$ 的表格表示

x	$x_0=a$	x_1	x_2	$\cdots$	x_i	x_{i+1}	$\cdots$	x_n
$y(x)$	y_0	y_1	y_2	$\cdots$	y_i	y_{i+1}	$\cdots$	y_n

使它满足方程：$\begin{cases} \dfrac{dy(x_i)}{dx}=g(x,y(x_i)),x_i\in[a,b] \\ y(a)=y_0 \end{cases}$　　（5.13）

不妨设自变量 x 的结点 $x_i(i=0,1,2,\cdots,n)$ 为 $a=x_0<x_1<x_2<\cdots<x_i<\cdots<x_n=b$，这些结点的分割是任意的。但为了方便起见，通常选取相邻两个结点间的距离 h 是等长的，即步长 $h=x_{i+1}-x_i$ 为定值，这时 $x_i=x_0+ih,(i=0,1,2,\cdots,n)$。

然后，从已知的 $x_0=a$，$y(a)=y(x_0)=y_0$ 点出发，设法求出 x_1 点的函数近似值 $y_1\approx y(x_1)$，再由已知的 y_0 和 y_1 求出 $y_2\approx y(x_2)$，以此类推，由 y_0，y_1，y_2 求出 y_3，…若能找出这样的递推公式（成为计算公式），就可以求出在自变量各个结点 x_i 上满足或近似满足式（5.13）的函数值 y_i，也就是求出了式（5.13）的数值解 $y_i\approx y(x_i)(i=0,1,2,\cdots,n)$。

可见，求一阶常微分方程初值问题数值解，就是把连续性方程（5.13）的求解变成求自变量在离散结点 x_i 上的函数近似值 $y_i\approx y(x_i)$。解决这个问题有多种方法，常用的有泰勒展开法、龙格–库塔法和阿达姆斯法等。

2．常微分方程初值问题数值解的 MATLAB 实现

在 MATLAB 中，有多个求解常微分方程初值问题数值解的指令，在此仅介绍两个常用的指令，想了解其他指令时，可用 help 调出“function(功能函数)库”进行查阅。

ode23 和 ode45 是求解常微分方程初值问题数值解的两个最常用的指令，指令中的 ode 是英文常微分方程“Ordinary Differential Equation”的缩写，它们都采用龙格库塔公式进行数值求解，23 和 45 分别表示使用的是 2/3 阶和 4/5 阶龙格库塔公式。它们的调用格式基本相同。在此仅以 ode23 为例作如下说明。指令 ode23 的调用格式如下：

[x,y]=ode23(‘fun’,tspan,y0,options)

①该指令适用于一阶微分方程组 $y_j'(x)=g_j(x,y_j),(j=1,2-\cdots)$，如遇到高阶微分方程，必须先把它变换为一阶常微分方程组，即状态方程，方可使用。

②输入参数“fun”为定义微分方程组 $\dfrac{\mathrm{d}y_j}{\mathrm{d}x}=g_j(x,y_j)$ 的 M-函数文件名，可以在文件名前加写@，或用英文格式单引号界定文件名。

③在编辑调试窗口中编写一阶常微分方程组 $y_j'=\dfrac{\mathrm{d}y_j}{\mathrm{d}x}=g_j(x,y_j)$ 的 M-函数文件时，每个微分方程的格式都必须与 $y_j'=g_j(x,y_j)$ 一致，即等号左端为待求函数 y_j 的一阶导数 $\dfrac{\mathrm{d}y_j}{\mathrm{d}x}$，右端函数的变量 x，y_j 严格以“先自变量 x、后函数 y_j”的固定顺序输入，y_j' 的下标 $j=1$，2，…表示微分方程的序数。

④输入参数“tspan”规定了常微分方程的自变量取值范围，它以矩阵[$x0,xf$]形式输入，表示自变量 $x\in[x0,xf]$。

⑤输入参数“y0”表示初始条件向量，$y0=[y(x0)\quad y'(x0)\quad y''(x0)\cdots]$。微分方程组中的方程个数必须等于初始条件数，这是求常微分方程特解所必需的条件。

⑥输入参数“options”表示选项参数，它可由 odeset 函数进行设置，较为复杂。

⑦输出参数[x,y]为微分方程组解函数的列表（x 和 y 都是列矩阵），它包含向量 x 各结点 x_i 和与 x_i 对应向量 y 的第 i 个分量 $y_i=y(x_i)$（即第 i 个方程解），i 表示结点序列数。

⑧输出参数[x,y]默认时输出解函数的曲线，即函数 $y=f(x)$ 及其各阶导数 $y_j=f^{(j)}(x)$ 的曲线。

除 ode23 外，求解微分方程的指令还有 ode45、ode113、ode15s、ode23s、ode23T 和 ode23TB。

例 5.29 求解 $\frac{\mathrm{d}y(x)}{\mathrm{d}x}+2xy(x)=1$，$y(-2.5)=0$。

解：该微分方程看似简单，若用符号法求解析解，输出结果含有无法确定取值的函数。若输入：

```
>> dsolve('Dy+2*x*y=1','y(-2.5)=0','x')
```

按回车键得出：

```
ans =
(-1/2*i*pi^(1/2)*erf(i*x)-1/2*i*pi^(1/2)*erf(5/2*i))*exp(-x^2)
```

函数 erf(i*x)无法取值，如果用数值解法，可按下述步骤进行。

（1）先在编辑调试窗中编出下列 M-函数文件：

```
%一阶微分方程 y'=1-2xy 的 M-函数文件
function y1=li5_0708(x,y)
y1=1-2*x*y;%一阶微分方程表达式
```

以“li5_3”为名存盘，退回到指令窗中。

（2）输入：

```
>> [x,y]=ode23(@li5_0708,[-2.5 3],[0])
```

按回车键可得出微分方程的数值解：

```
x =
   -2.5000
   -2.4999
   -2.4995
   -2.4975
    ......
    2.6758
    2.7884
    2.9154
    3.0000
y =
         0
    0.0001
    0.0005
    0.0025
    ......
    0.2876
    0.2389
    0.2055
    0.1910
```

输入：

```
>> size(x)
```

按回车键得出：

```
ans =
```

```
     47     1
```

表明 x 被离散分成 47 个结点，对应地得出 y 的 47 个取值，即微分方程的数值解。

（3）若输入：

```
>> ode23(@li5_0708,[-2.5 3],[0]),grid
```

按回车键得如图 5-15 所示微分方程的图示解。

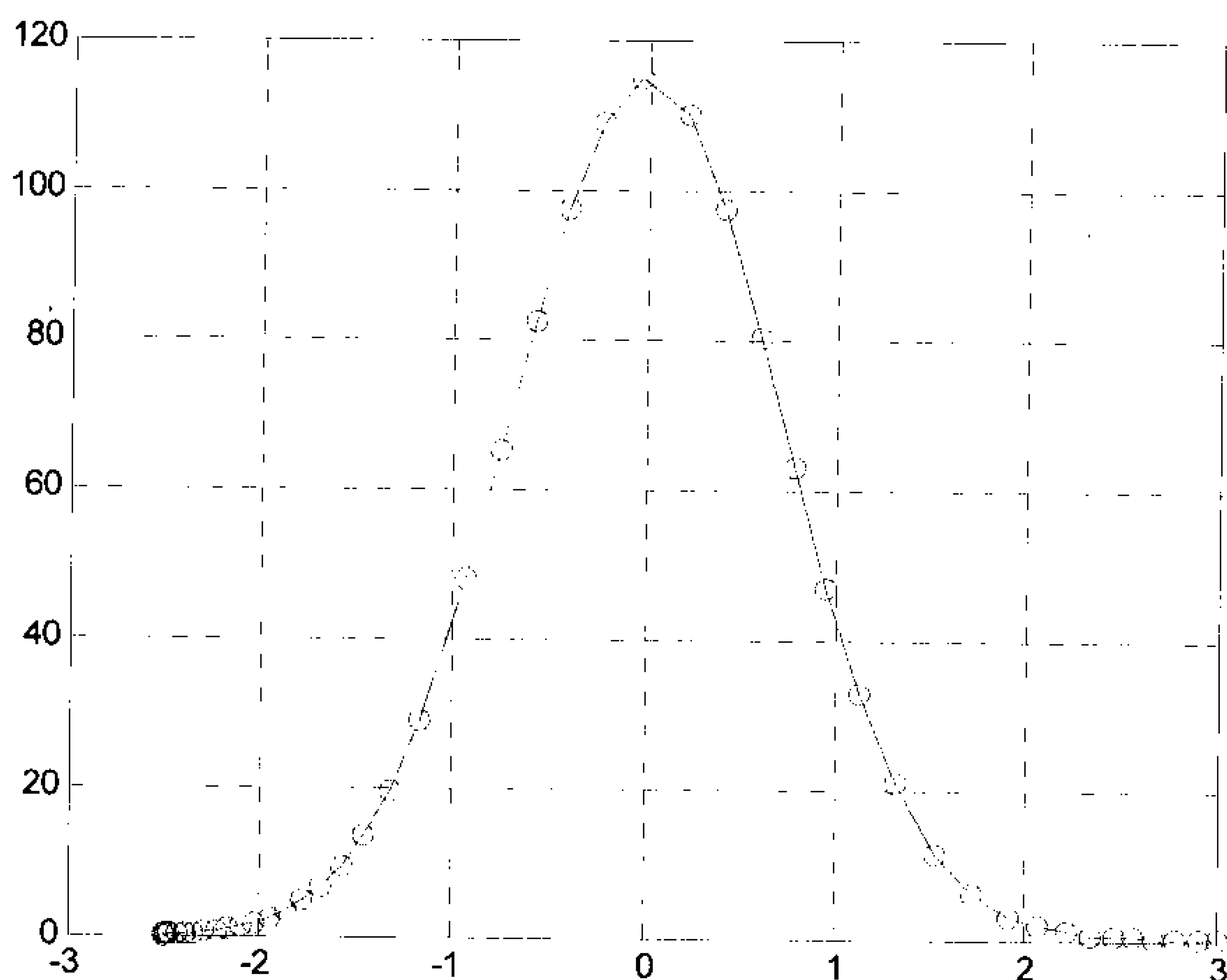


图 5-15 微分方程图示解

由例 5.29 可以看出使用 ode23 指令时写与不写输出变量的差异。常微分方程初值问题的数值解，就是求出解函数的列表法或图示法表示，输出数值是解函数的列表法表示，输出曲线是解函数的图示法表示。

例 5.30 求解 $\dfrac{\mathrm{d}^2 y}{\mathrm{d}t^2}+y=1-\dfrac{t^2}{\pi}$，满足 $y(-2)=-5$，$y'(-2)=5$，$t\in[-2,7]$。

解：①把二阶常微分方程变换成两个一阶常微分方程组成的微分方程组。

令 $y_1=y(t)$，$y_2=\dfrac{\mathrm{d}y_1}{\mathrm{d}t}$，则原微分方程转化为下面的微分方程组：

$$\begin{cases}\dfrac{\mathrm{d}y_1}{\mathrm{d}t}=y_2\\[2mm]\dfrac{\mathrm{d}y_2}{\mathrm{d}t}=-y_1+1-\dfrac{t^2}{\pi}\end{cases}$$

该方程组也可以写成矩阵方程。

设 $Y=\begin{bmatrix}y_1\\y_2\end{bmatrix}$，则微分方程组可以写成：

$$\frac{\mathrm{d}Y}{\mathrm{d}t}=\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}y_1\\y_2\end{bmatrix}=\begin{bmatrix}0&1\\-1&0\end{bmatrix}\begin{bmatrix}y_1\\y_2\end{bmatrix}+\begin{bmatrix}0\\1\end{bmatrix}(1-\frac{t^2}{\pi})=\begin{bmatrix}0&1\\-1&0\end{bmatrix}Y+\begin{bmatrix}0\\1\end{bmatrix}(1-\frac{t^2}{\pi})$$

②在编辑调试窗口中建立关于微分方程的 M-函数文件。可写成下述两种形式之一。

a.建立和微分方程组对应的形式，可输入：

```
function dy=li5_0709(t,y) %定义输入、输出变量和函数文件名
dy=zeros(2,1); %确定 dy 的维数，用微分方程组时不可默认
```



```
dy(1)=y(2); %dy(m)表示y的m阶导数；y(n)表示y的第n列
dy(2)=-y(1)+1-t^2/pi; %与方程组中第二个微分方程相对应
```

注意：此题 $y(n)$ 表示输出时 y 的第 n 列（也是微分方程组的第 n 个方程中的解函数）：当 n=1 时，y 的第一列表示函数 $y(t)$；当 n=2 时，y 的第二列表示函数 $y(t)$ 的一阶导数 $y'=\frac{dy_1}{dt}$。

b.建立和矩阵方程对应的形式，可输入：

```
function dY=li5_0709_1(t,Y)%定义变量t,Y,dY和文件名li5_0709_1
dY=[0 1;-1 0]*Y+[0;1]*(1-t^2/pi);%与矩阵方程形式对应
```

③把编好的M-函数文件分别以"li5_0709"和"li5_0709_1"为名存盘，回到指令窗。

④在指令窗中使用ode23，可根据需要用下述两种方法之一求解微分方程。

a.不写输出参量，得出微分方程的图示解，输入：

```
>> ode23(@li5_0709,[-2 7],[-5 5]),grid
```

按回车键得出如图5-16所示曲线，不输出数据。

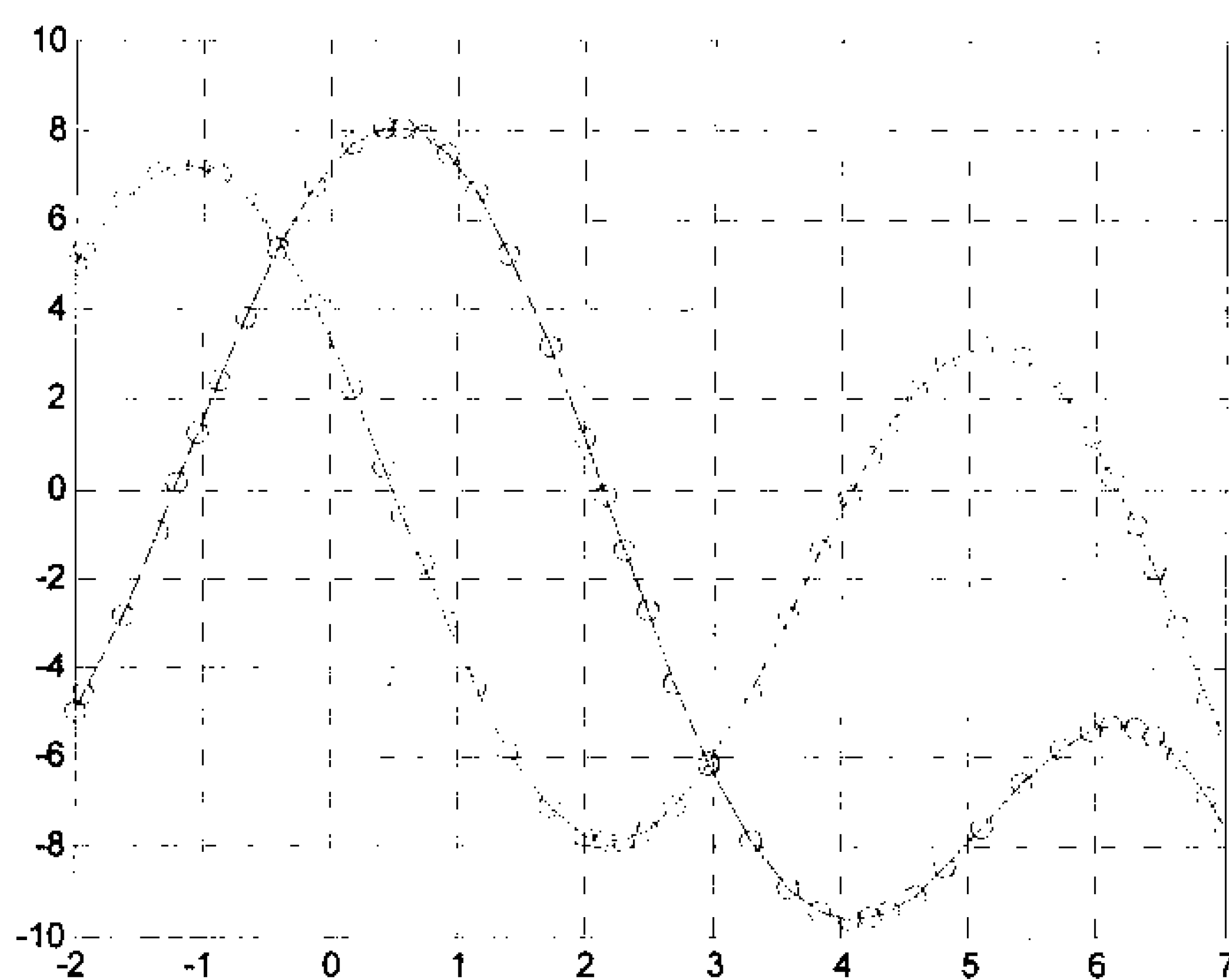


图5-16　例5.30中二阶微分方程的一条特解函数及其导函数曲线

b.写有输出参量，得出微分方程的数值解，输入：

```
>> [t,y]=ode23(@li5_0709,[-2 7],[-5 5])
```

按回车键将显示出自变量 t 和两个待求函数 Y(:,1)=y(t)和 Y(:,2)=y'(t)的对应数据。为节省篇幅，省写了许多数据，用"……"代替了：

```
t =
   -2.0000
   -1.9200
    ……
    6.8681
    7.0000
y =
   -5.0000    5.0000
   -4.5852    5.3658
    ……
   -6.8931   -4.7083
   -7.5759   -5.6421
```

再输入：

```
>> s1=size(t),s2=size(y)
```

按回车键得出：

```
s1 =
     42     1
s2 =
     42     2
```

表明自变量被分为 42 个结点，并计算出了对应点上的函数值及其一阶导数值。

例 5.31 求解微分方程 $x^3\dfrac{d^3y}{dx^3}+x^2\dfrac{d^2y}{dx^2}-4x\dfrac{dy}{dx}=3x^2$ 的解，使其满足初始条件：$y(1)=0$，$y'(1)=-1$，$y''(1)=1$的解。

解：①首先将三阶常微分方程变换成 3 个一阶常微分方程组成的微分方程组。为此，令：

$$y_1=y(x),\quad y_2=\frac{dy(x)}{dx}=\frac{dy_1}{dx},\quad y_3=\frac{d^2y(x)}{dx^2}=\frac{dy_2}{dx}$$

于是可得微分方程组：

$$\begin{cases}\dfrac{dy_1}{dx}=y_2\\ \dfrac{dy_2}{dx}=y_3\\ \dfrac{dy_3}{dx}=-\dfrac{y_3}{x}+\dfrac{4y_2}{x^2}+\dfrac{3}{x}\end{cases}$$

②打开编辑调试窗，编辑与微分方程组对应的 M-函数文件。

```
function dy=li5_07101(x,y)
dy=zeros(3,1); %规定变量 dy 的维数，使用微分方程组时，不可缺少
dy(1)=y(2); %dy(1)表示 y 的一阶导数；y(2)表示 y 的第二列，即 y'(x)
dy(2)=y(3); %dy(2)表示 y 的二阶导数；y(3)表示 y 的第三列，即 y''(x)
dy(3)=-y(3)/x+4*y(2)/x^2+3/x;
```

以“li5_07101”为函数文件名存盘。

③在指令窗中可根据需要选择下述两种方法之一求解微分方程。

a.如果输入：

```
>> ode23(@li5_07101,[1 4],[0 -1 1]),grid
```

按回车键得出如图 5-17 所示。

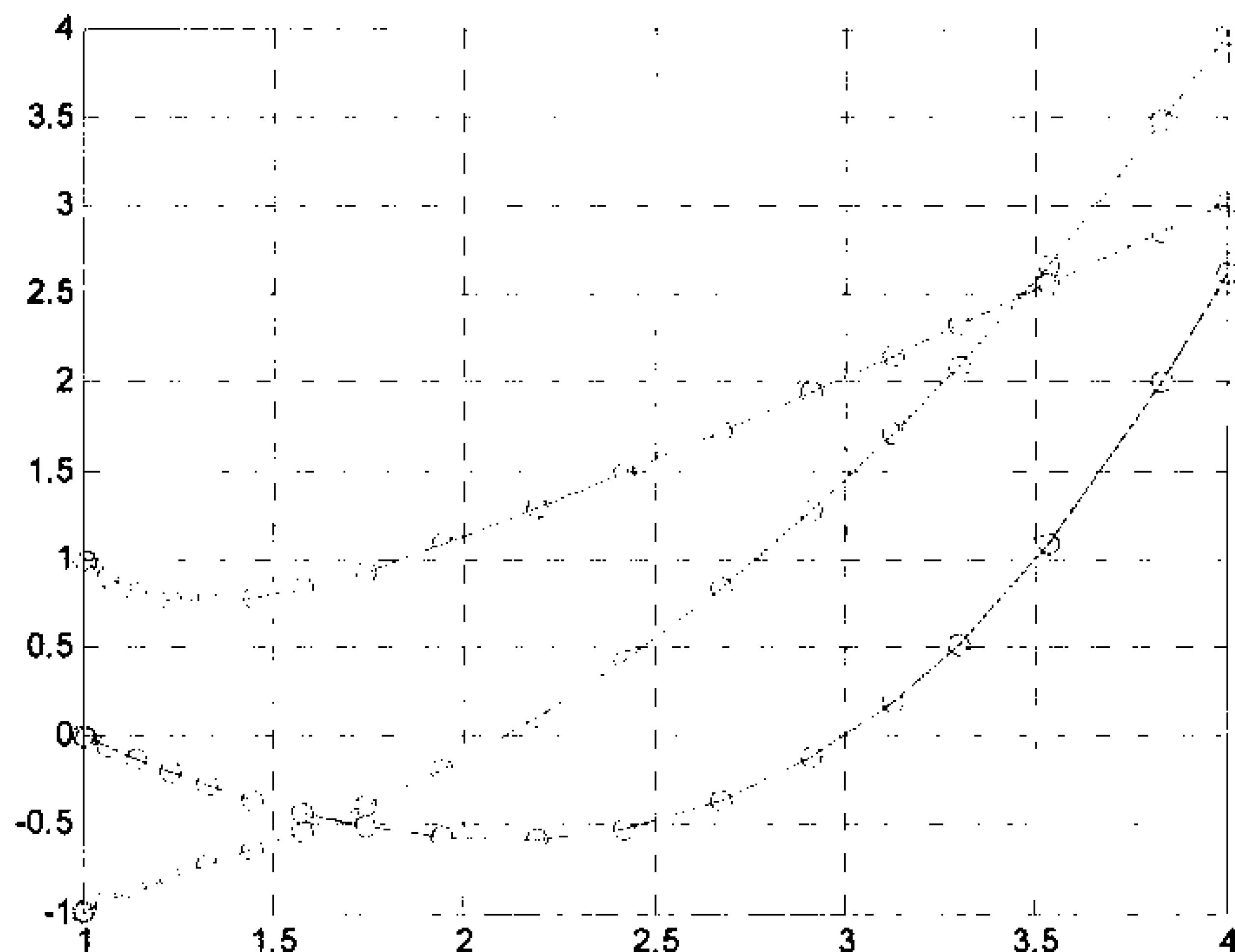


图 5-17　例 5.31 中三阶微分方程的一条特解函数及其一、二阶导函数曲线

由于没写输出变量，只输出曲线，没有数据。

b.如果输入：

```
>> [x,y]=ode23(@li5_07101,[1 4],[0 -1 1])
```

按回车键得出下列数据（中间省略了一些数据）：

```
x =
    1.0000
    1.0001
    1.0005
    ……
    3.8321
    4.0000
y =
         0    -1.0000     1.0000
   -0.0001    -0.9999     0.9998
   -0.0005    -0.9995     0.9990
    ……
    1.9984     3.4746     2.8491
    2.6228     3.9670     3.0148
```

输入：

```
>> xn=size(x),yn=size(y)
```

按回车键得出：

```
xn =
    22     1
yn =
    22     3
```

表明 x 有 22 个结点，y(:,j)(j=1,2,3)分别为 y(x)、一阶导数 $y'(x)$ 和二阶导数 $y''(x)$ 在每个结点上的取值。

作为综合练习，可输入下述指令，并仔细理解每步程序的意义：

```
>> [x,y]=ode23(@li5_07101,[1 4],[0 -1 1]);
>> poly2str(polyfit(x,y(:,1),3),'t')
```

按回车键得出：

```
ans =
   0.14176 t^3 - 0.26304 t^2 - 0.78818 t + 0.90515
```

若用符号法求解三阶微分方程，可输入：

```
>> dsolve('t^3*D3y+t^2*D2y-4*t*Dy=3*t^2','y(1)=0,Dy(1)=-1,D2y(1)=1','t')
```

为了绘图，输入：

```
>> t=1:0.2:4;
>> y10=0.14176*t.^3 - 0.26304*t.^2 - 0.78818*t + 0.90515;
>> y11=1/6*t.^3-1/2*t.^2+1/2./t-1/6;
>> plot(t,y10,'O',t,y11,'r+',x,y(:,1)),legend('拟合点','数值解','符号解')
```

按回车键得出如图 5-18 所示曲线。

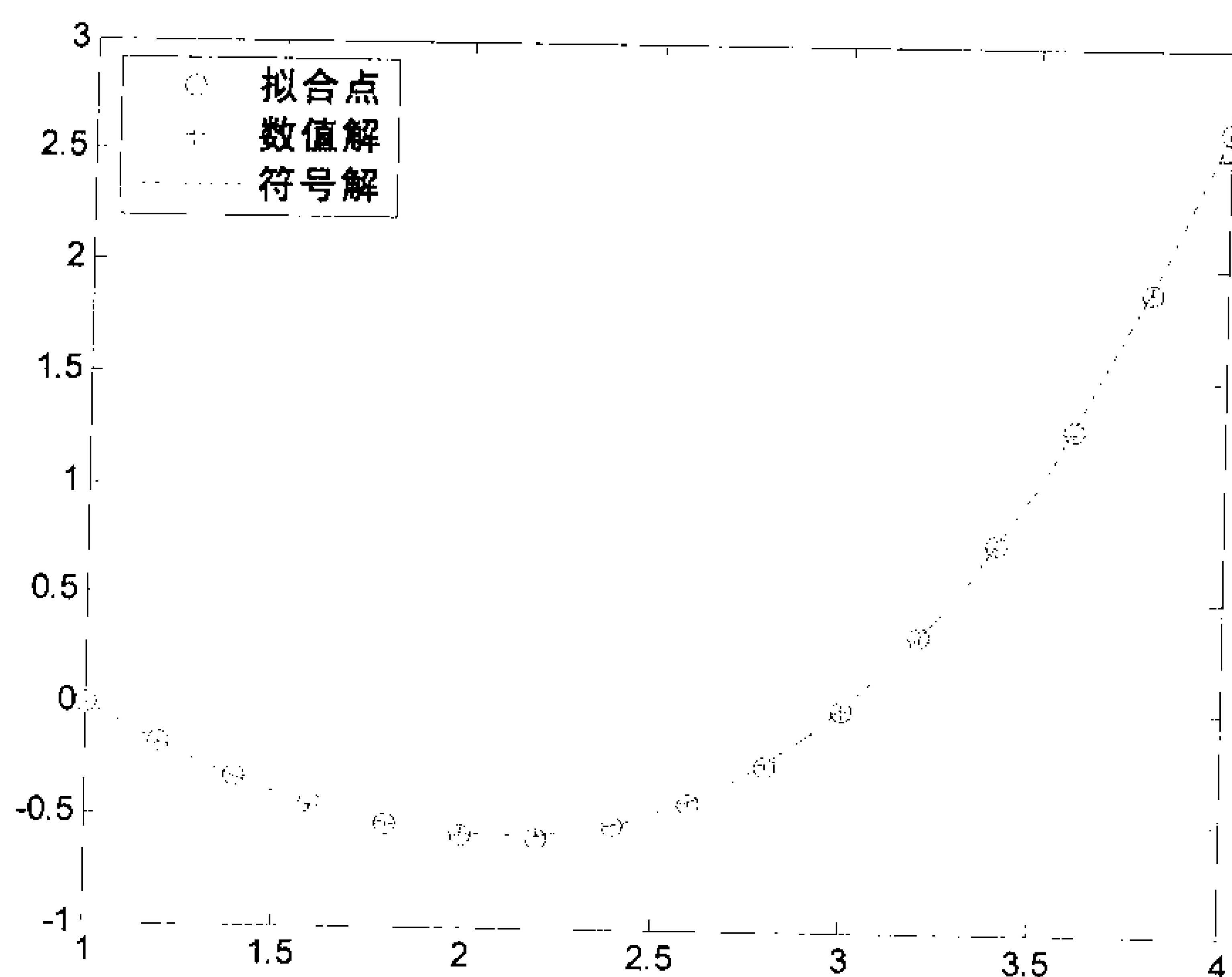


图 5-18 例 5.31 中三阶微分方程的一条特解函数曲线

5.7 求解偏微分方程

现代科学技术工程中的大量数学模型都可以用微分方程来描述，很多近代自然科学的基本方程本身就是微分方程。微分方程，特别是偏微分方程的定解问题几乎成了表述自然与工程技术领域中各种现象最重要的数学工具，应用十分广泛，尤其在高、精、深及最前沿的科学领域，几乎时时处处都在和偏微分方程打交道。但无奈的是，绝大多数偏微分方程都不能求得其实用的解析解，因此，应用计算机得到其数值（近似）解成了唯一解决问题的途径。

鉴于偏微分方程数值解在数学计算中越来越重要的地位，本节介绍 MATLAB 中专门用来求解偏微分方程的又一功能强大的软件包——PDE Toolbox。由于篇幅所限及偏微分方程解法本身的复杂性，本节仅对一些简单、基本的经典偏微分方程给出其求解方法。

5.7.1 偏微分方程组求解

MATLAB 提供了 pdepe()函数，可以直接求解偏微分方程组。首先，将偏微分方程组转换为下面的形式：

$$c(x,t,u,\frac{\partial u}{\partial x})\frac{\partial u}{\partial t}=x^{-m}\frac{\partial}{\partial x}\left[x^{m}f(x,t,u,\frac{\partial u}{\partial x})\right]+s(x,t,u,\frac{\partial u}{\partial x}) \qquad (5.14)$$

这样，偏微分方程可以编写一个如下的 MATLAB 函数描述为

$$[c,f,s]=\text{pdefun}(x,t,u,u_x)$$

其中，pdefun 为函数名。这样，由给定的输入变量即可计算出 c，f，s 这 3 个函数。

边界条件可以用下面的函数描述为：

$$p(x,t,u)+q(x,t,u).*f(x,t,u,\frac{\partial u}{\partial x})$$

这样的边值函数可以编写一个如下的 MATLAB 函数描述为：

$$[p_a,q_a,p_b,q_b]=\text{pdebc}(x,t,u,u_x)$$

除了这两种函数外，还应写出初始条件函数。偏微分方程初始条件的数学描述为 $u(x,t_0)=u_0$。这样，需要一个简单的函数来描述，编写简单函数 $u_0=\text{pdeic}(x)$ 即可。

还可以选择 x 和 t 的向量，再加上描述的这些函数，就可以用 pdepe()函数求解偏微分方程，这需要用下面的格式求解该偏微分方程：

sol=pdepe(m,@pdefun,@pdeic,@pdebc,x,t)

例 5.32 试求解下面的偏微分方程：

$$\begin{cases}\dfrac{\partial u_1}{\partial t}=0.024\dfrac{\partial^2 u_1}{\partial x^2}-F(u_1-u_2)\\ \dfrac{\partial u_2}{\partial t}=0.17\dfrac{\partial^2 u_2}{\partial x^2}+F(u_1-u_2)\end{cases}$$

其中，$F(x)=\mathrm{e}^{5.73x}-\mathrm{e}^{-11.46x}$，且满足初始条件 $u_1(x,0)=1$，$u_2(x,1)=0$ 及边界条件 $\dfrac{\partial u_1}{\partial x}(0,t)=0$，$u_1(1,t)=1$，$\dfrac{\partial u_2}{\partial x}(1,t)=0$，$u_2(0,t)=0$。

解：对照给出的偏微分方程和式（5.14），则可以将原方程改写为：

$$\begin{bmatrix}1\\1\end{bmatrix}.*\frac{\partial}{\partial t}\begin{bmatrix}u_1\\u_2\end{bmatrix}=\frac{\partial}{\partial x}\begin{bmatrix}0.024\dfrac{\partial u_1}{\partial x}\\0.17\dfrac{\partial u_2}{\partial x}\end{bmatrix}+\begin{bmatrix}-F(u_1-u_2)\\F(u_1-u_2)\end{bmatrix}$$

可见，m=0，且：

$$c=\begin{bmatrix}1\\1\end{bmatrix},\ f=\begin{bmatrix}0.024\dfrac{\partial u_1}{\partial x}\\0.17\dfrac{\partial u_2}{\partial x}\end{bmatrix},\ s=\begin{bmatrix}-F(u_1-u_2)\\F(u_1-u_2)\end{bmatrix}$$

这样，可以编写出下面的描述偏微分方程的 MATLAB 函数为：

```
function [c,f,s]=c7mpde(x,t,u,du)
c=[1;1];y=u(1)-u(2);F=exp(5.73*y)-exp(-11.46*y);s=F*[-1;1];
f=[0.024*du(1);0.17*du(2)];
```

套用式（5.14）中的边界条件，可以写出如下的边值方程：

$$\text{左边界}\begin{bmatrix}0\\u_2\end{bmatrix}+\begin{bmatrix}1\\0\end{bmatrix}.*f=\begin{bmatrix}0\\0\end{bmatrix},\ \text{右边界}\begin{bmatrix}u_1-1\\0\end{bmatrix}+\begin{bmatrix}0\\1\end{bmatrix}.*f=\begin{bmatrix}0\\0\end{bmatrix}$$

从而编写出下面的描述边界条件的 MATLAB 函数：

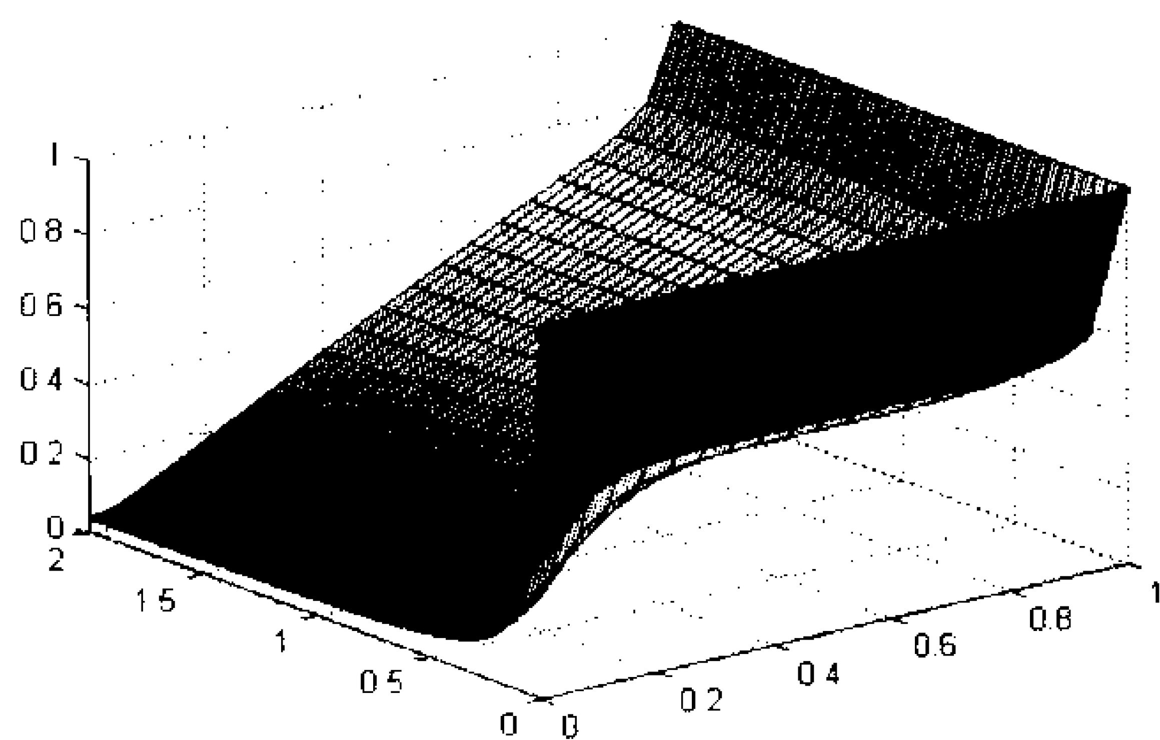
```
function [pa,qa,pb,qb]=c7mpbc(xa,ua,xb,ub,t)
pa=[0;ua(2)];qa=[1;0];pb=[ub(1)-1;0];qb=[0;1];
```

另外，还可以立即得出描述初值的 MATLAB 函数为：

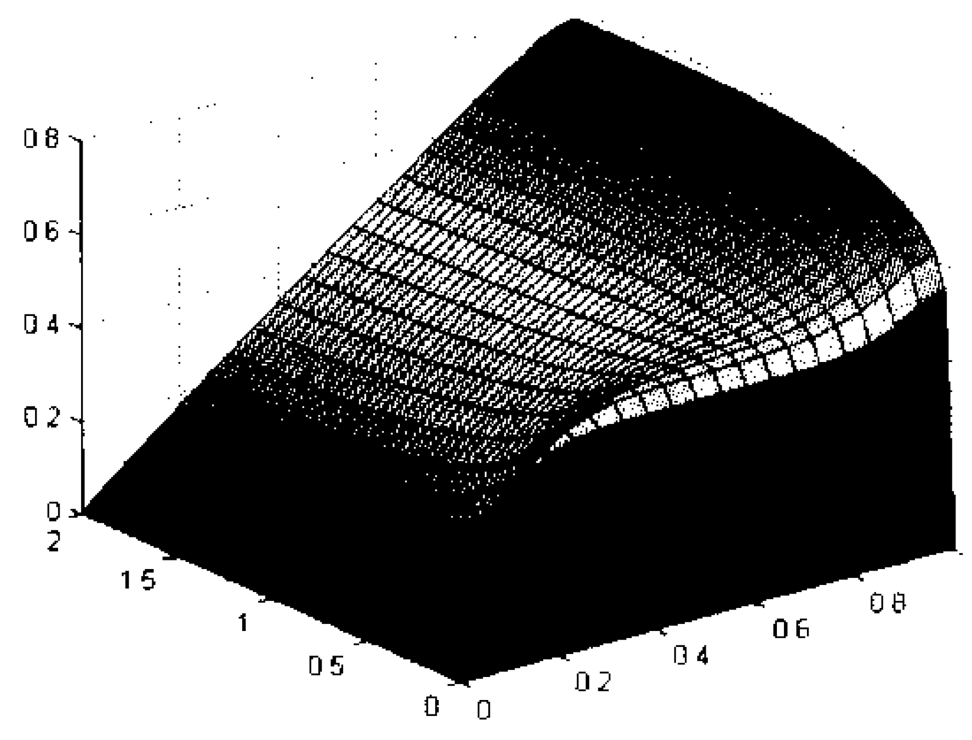
```
function u0=c7mpic(x)
u0=[1;0];
```

有了这 3 个函数，选定 x 和 t 向量，则可以由下面的语句直接求解此偏微分方程，得出解 u_1 和 u_2，如图 5-19（a）和图 5-19（b）所示。在 MATLAB 文本编辑窗口编制 M 文件：

```
>> x=0:0.05:1;t=0:.05:2;m=0;
>> sol=pdepe(m,@c7mpde,@c7mpic,@c7mpbc,x,t);
>> surf(x,t,sol(:,:,1))
>> figure
>> surf(x,t,sol(:,:,2))
```



（a） $u_1(t,x)$



（b） $u_2(t,x)$

图 5-19 偏微分方程求解区域设置

5.7.2 二阶偏微分方程的数学描述

除了前面介绍的一阶偏微分方程外，MATLAB 语言还有自己的偏微分方程工具箱，可以比较规范地求解各种常见的二阶偏微分方程。这里将 MATLAB 偏微分方程工具箱可解的二阶偏微分方程简单介绍一下，后面再介绍一个实用的偏微分方程求解界面 pdetool，利用该程序界面可以容易地求解偏微分方程。

1．椭圆形偏微分方程

椭圆形偏微分方程的一般表示形式为：

$$-\text{div}(c\nabla u)+au=f(x,t)$$

其中，若 $u=u(x_1,x_2,\cdots,x_n)=u(x,t)$， ∇u 为 u 的梯度，则其定义为：

$$\nabla u=\left[\frac{\partial}{\partial x_1},\frac{\partial}{\partial x_2},\cdots,\frac{\partial}{\partial x_n}\right]u$$

散度 div(v)的定义为：

$$\text{div}(v)=\left(\frac{\partial}{\partial x_1}+\frac{\partial}{\partial x_2}+\cdots+\frac{\partial}{\partial x_n}\right)v$$

这样，div($c\nabla u$)可以更明确地表示成：

$$\text{div}(c\nabla u)=\left[\frac{\partial}{\partial x_1}\left(c\frac{\partial u}{\partial x_1}\right)+\frac{\partial}{\partial x_2}\left(c\frac{\partial u}{\partial x_2}\right)+\cdots+\frac{\partial}{\partial x_n}\left(c\frac{\partial u}{\partial x_n}\right)\right]$$

若 c 为常数，则该项可以进一步简化为：

$$\operatorname{div}(c\nabla u)=c\left(\frac{\partial^2}{\partial x_1^2}+\frac{\partial^2}{\partial x_2^2}+\cdots+\frac{\partial^2}{\partial x_n^2}\right)u=c\Delta u$$

其中，Δ 又成为 Laplace 算子。这样，椭圆形微分方程可以简单地写出：

$$-c\left(\frac{\partial^2}{\partial x_1^2}+\frac{\partial^2}{\partial x_2^2}+\cdots+\frac{\partial^2}{\partial x_n^2}\right)u+au=f(x,t)$$

2．抛物线型偏微分方程

抛物线型偏微分方程的一般表示形式为：

$$d\frac{\partial u}{\partial t}-\operatorname{div}(c\nabla u)+au=f(x,t)$$

根据上面的叙述，若 c 为常数，则该方程可以更简单地写成：

$$d\frac{\partial u}{\partial t}-c\left(\frac{\partial^2 u}{\partial x_1^2}+\frac{\partial^2 u}{\partial x_2^2}+\cdots+\frac{\partial^2 u}{\partial x_n^2}\right)+au=f(x,t)$$

3．双曲型偏微分方程

双曲型偏微分方程的一般表示形式为：

$$d\frac{\partial^2 u}{\partial t^2}-\operatorname{div}(c\nabla u)+au=f(x,t)$$

若 c 为常数，则该方程可以更简单地写成：

$$d\frac{\partial^2 u}{\partial t^2}-c\left(\frac{\partial^2 u}{\partial x_1^2}+\frac{\partial^2 u}{\partial x_2^2}+\cdots+\frac{\partial^2 u}{\partial x_n^2}\right)+au=f(x,t)$$

从上面所述的 3 种类型方程可以看出，它们直接的区别在于 u 函数对 t 的导数阶次。如果对 t 没有求导，则可以理解为其值为常数，故称为椭圆形偏微分方程。如果取 u 对时间的一阶导数，则一阶导数与 u 对 x 的二阶导数直接构成了抛物线关系，故称其为抛物线型偏微分方程。如果对 t 取二阶导数，则可以称之为双曲型偏微分方程。

MATLAB 的偏微分方程工具箱采用的是有限元方法求解各种偏微分方程的。椭圆形偏微分方程求解中，c，a，d，f 均可以为给定函数的形式，但其他类型偏微分方程求解时，它们必须为常数。

4．特征值型偏微分方程

MATLAB 可以直接求解的另一类偏微分方程为特征值型偏微分方程，其一般表示形式为：

$$-\operatorname{div}(c\nabla u)+au=\lambda du$$

对常数 c，该方程还可以简化成：

$$-c\left(\frac{\partial^2 u}{\partial x_1^2}+\frac{\partial^2 u}{\partial x_2^2}+\cdots+\frac{\partial^2 u}{\partial x_n^2}\right)+au=\lambda du$$

对比式（5-）和（5-）可以发现，将前者等号右侧的 λdu 移动到方程的左侧，就可以变换成一般的椭圆形偏微分方程，所以该方程是椭圆形偏微分方程的一个特例。

5.7.3 偏微分方程求解界面简介

1．偏微分方程求解程序概述

MATLAB 偏微分方程工具箱提供了一个界面，可以求解二元偏微分方程$u(x_1,x_2)$，这时求解区域可以用该界面提供的画圆、椭圆、矩形及多边形等工具任意绘制，也可以由若干个这样简单绘制的集合进行并集、交集、差集等构成所需的求解区域。完成求解区域的绘制后，还可以用该界面提供的功能将原求解区域用三角形的形式自动绘制出网格。

在 MATLAB 命令提示符下输入 pdetool，将启动偏微分方程求解界面，如图 5-20 所示。

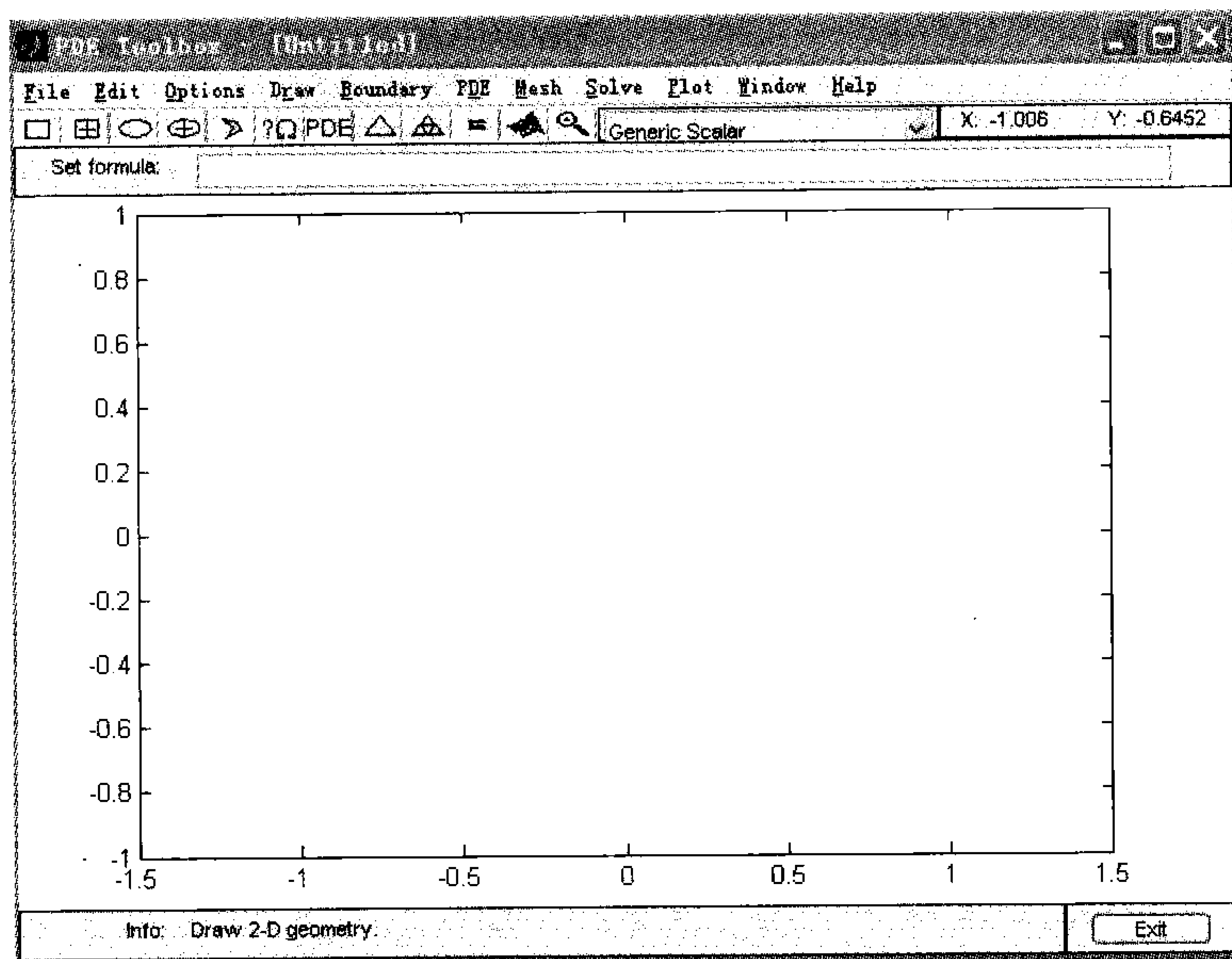


图 5-20 偏微分方程求解界面

偏微分方程求解界面分为如下几个部分。

1）菜单系统

偏微分工具箱有较全面的菜单系统，其中大部分实用功能均可以由工具栏实现，工具栏不能实现的部分多为一些工具箱的设置与文件处理的功能。后面将根据实际需要介绍菜单系统的若干功能。

2）工具栏

工具栏内各个按钮的功能如图 5-21 所示，工具栏能实现从求解区域设定、微分方程参数描述、求解到结果表示在内的一整套实际功能。工具栏右侧的列表框还给出了 MATLAB 能直接求解的一些常用微分方程类型。

3）集合编辑（Set formula）

用户可以在求解区域用不同的几何形状画出若干集合，而集合编辑区域允许用户用加减法等表示集合的并、交和差集运算，更精确地描述求解区域。

4）求解区域

为该程序界面下部的区域，用户可以在这个部分内绘制出问题的求解区域，微分方程的解也可以在这个区域内用二维的形式表示出来。MATLAB 还支持三维表示，但需要打开新的图形窗口。

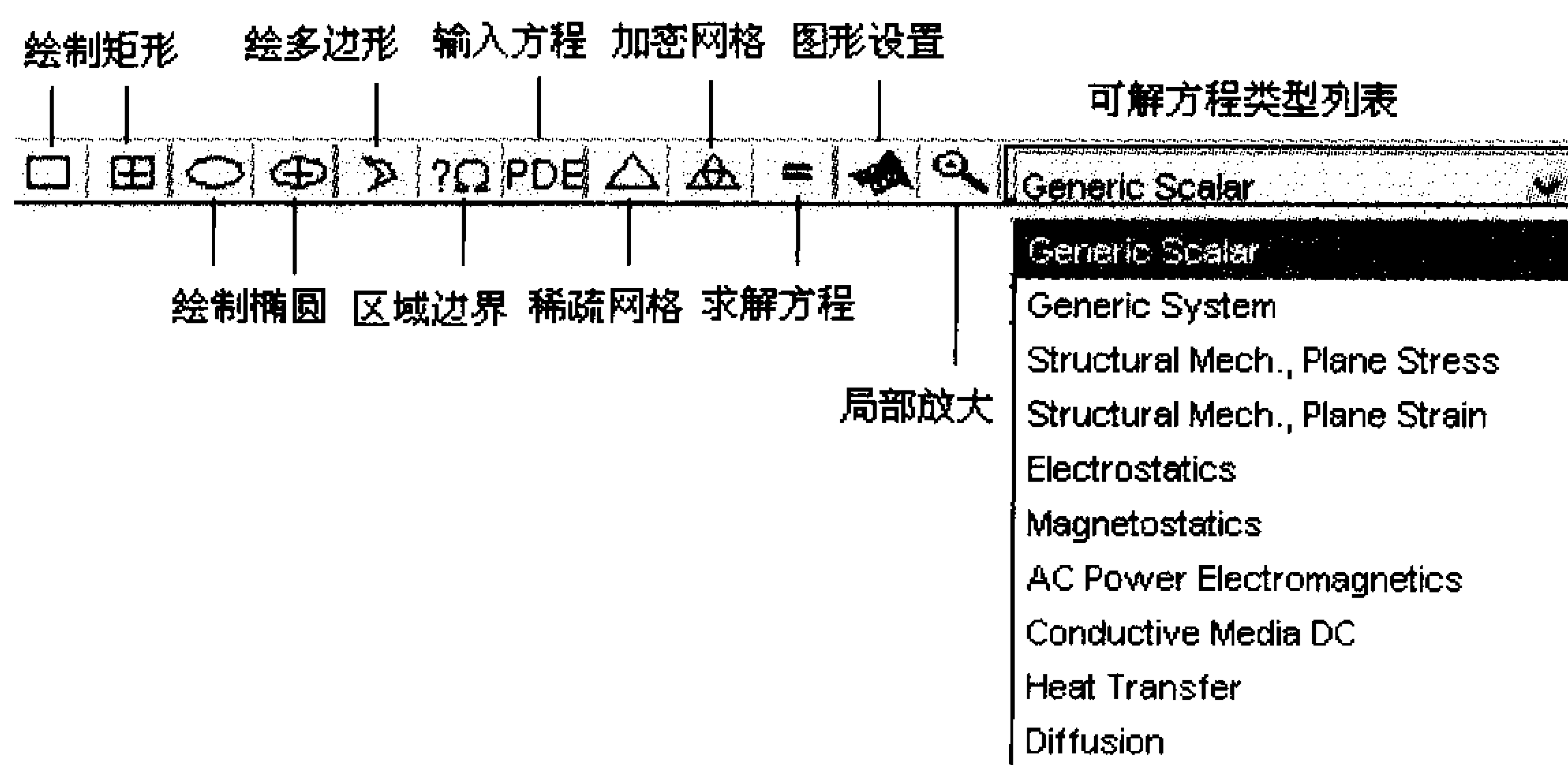
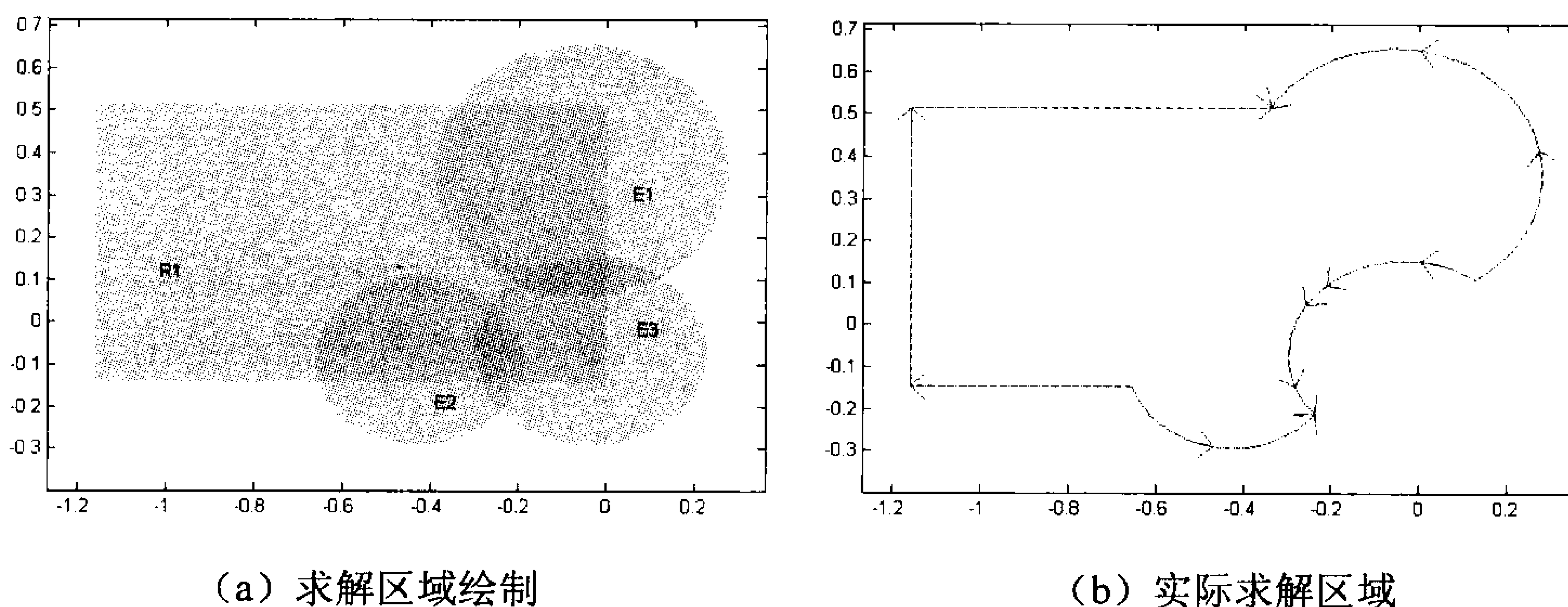


图 5-21　偏微分方程求解工具栏

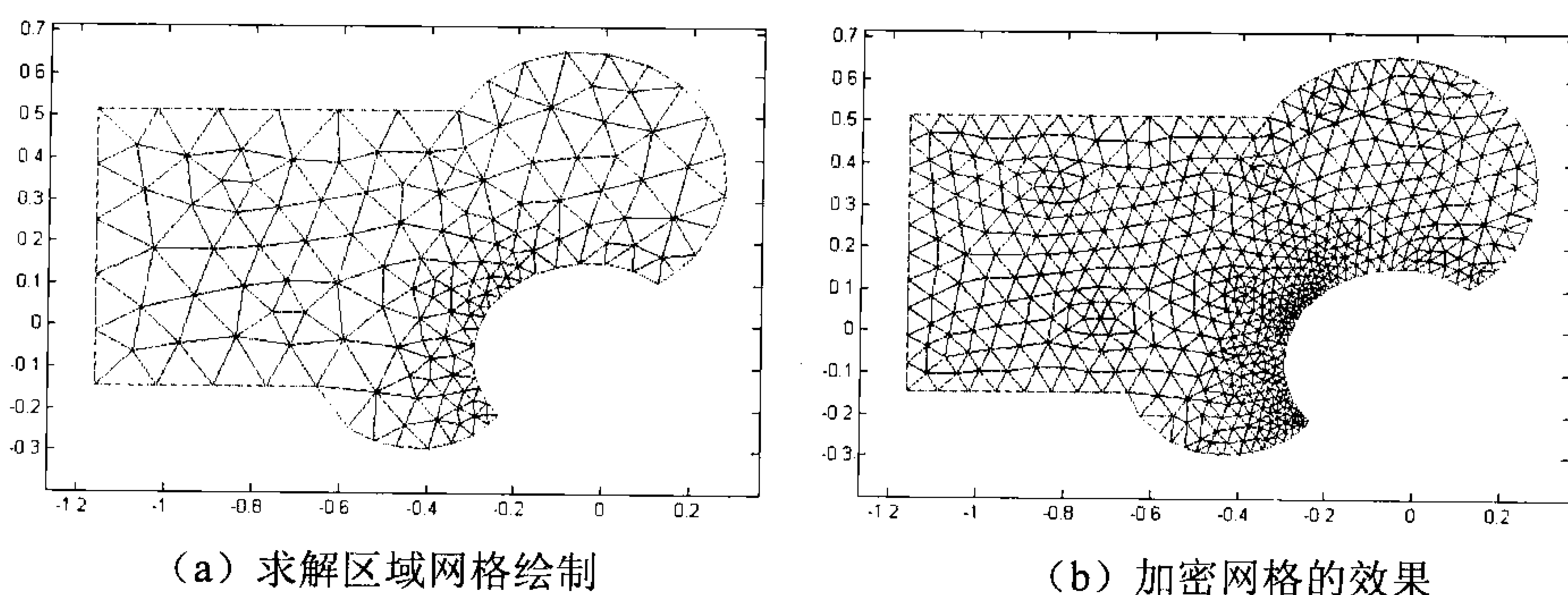
2．偏微分方程求解区域绘制

下面将通过例子演示在偏微分方程求解界面下描述求解区域的方法。首先用工具栏中提供的椭圆绘制和矩形绘制功能绘制出如图 5-22（a）所示的一些区域，这样就可以在集合编辑栏目中将原来的内容修改为(R1+E1+E2)-E3，表示从矩形 R1，椭圆 E1，E2 的并集中剔除掉 E3。单击工具栏中的 $\partial\Omega$ 图标就可以得到求解区域。执行【Boundary】→【Remove All Subdomain】→【Borders】命令，则将消除若干相邻区域中间的分割线，得出如图 5-22（b）所示的区域图。有了求解区域，就可以单击△图标将求解区域用三角形划分成若干网格，如图 5-23（a）所示。如果感觉到网格不够密，则可以单击右侧的按钮加密网格，可以得出如图 5-23（b）所示的更密的网格图。值得指出的是，一般情况下，网格越密，计算的结果越精确，但代价是计算时间也越长。



（a）求解区域绘制　　（b）实际求解区域

图 5-22　偏微分方程求解区域设置



（a）求解区域网格绘制　　　　（b）加密网格的效果

图 5-23　求解区域的网格生成

3．偏微分方程边界条件描述

求解边界在偏微分方程界面下用 $\partial\Omega$ 表示。一般，在偏微分方程工具箱支持的边界条件包括 Dirichlet 和 Neumann 条件。下面分别介绍这两种边界条件。

1）Dirichlet 条件

一般描述为：

$$h\left(x,t,u,\frac{\partial u}{\partial x}\right)u|\partial\Omega = r\left(x,t,u,\frac{\partial u}{\partial x}\right)$$

其中，$\partial\Omega$ 表示求解区域的边界。假设在边界上满足该方程，则只需给出 r 和 h 函数即可，这两个参数可以为常数，也可以为 x 的函数，甚至可以是 $u,\dfrac{\partial u}{\partial x}$ 的函数，为方便起见，一般可以令 h=1。后面将介绍 Dirichlet 边界条件的描述方式。

2）Neumann 条件

其扩展形式为：

$$\left[\frac{\partial}{\partial n}(c\nabla u)+qu\right]|\partial\Omega = g$$

其中，$\partial u/\partial n$ 为 x 向量法向的偏导数。

执行【Boundary】→【Specify Boundary Condtions】命令，将打开一个如图 5-24 所示的对话框，用户可以在这个对话框中描述边界条件。如果想使得边界上各点的函数值为 0，则可以将该对话框的“r”栏值设为 0 即可。

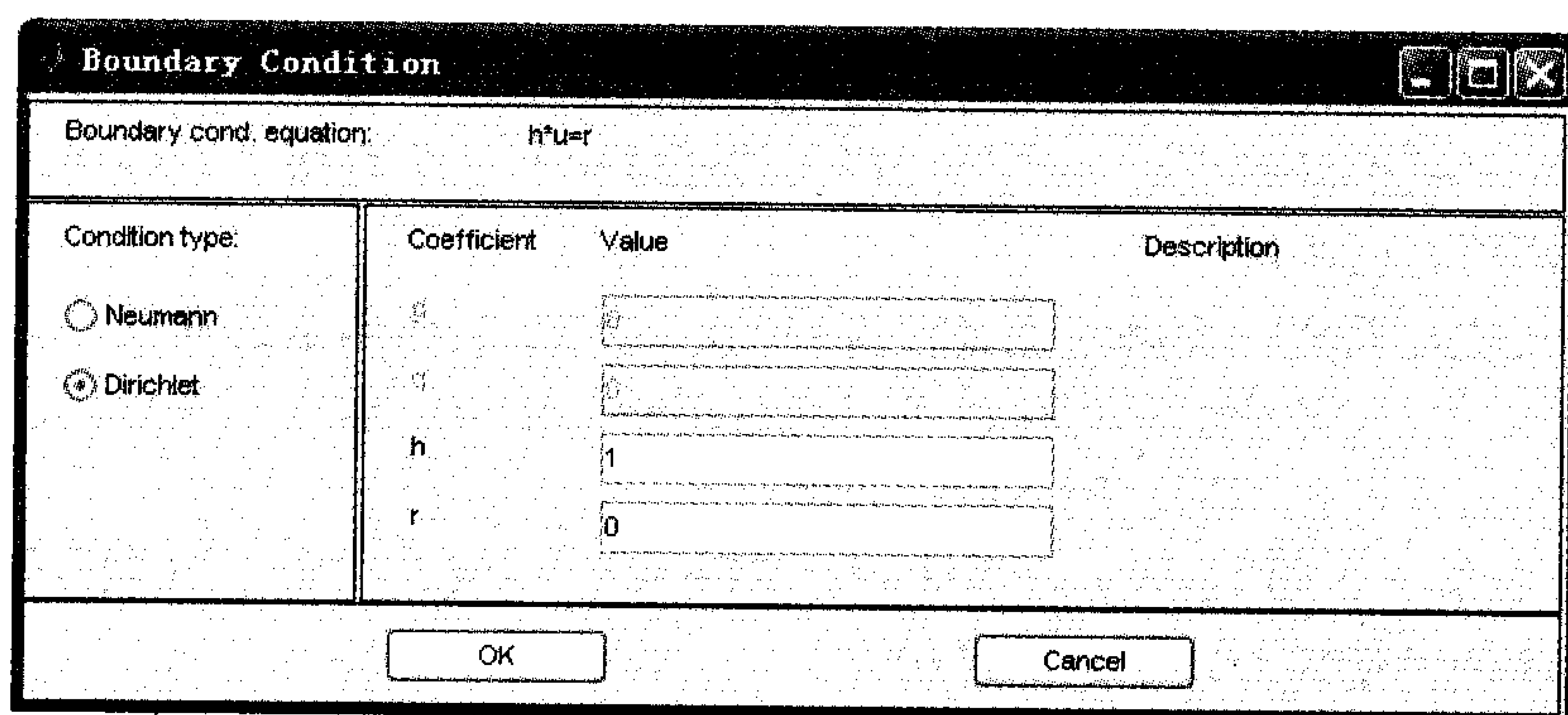


图 5-24　边界条件设置对话框

用前面的方法设置了求解区域和边界条件，并选择了合适的偏微分方程后，就可以单击工具栏的等号图标立即得出微分方程的解。

5.8 最优化问题

最优化理论和方法是第二次世界大战之后迅速发展起来的一门新兴学科，它具有很强的实践性，被广泛应用于科学计算、工程设计、管理决策、商业操作和军事指挥等各个领域。如今，它已成为具有多分支的综合学科。计算机技术的飞速发展，为最优化方法的实施提供了有力的工具，促进了最优化理论和方法的发展，推动了最优化方法的广泛应用。

所谓最优化就是找出使得目标函数值达到最小或最大的自变量值的方法。从其分类看有无约束最优化问题和有约束最优化问题。

5.8.1 无约束最优化问题求解

无约束最优化问题是最简单的一类最优化问题，其一般数学描述为：

$$\min_x f(x)$$

其中，$x=[x_1,x_2,\cdots,x_n]^T$ 称为优化变量，$f(x)$ 函数称为目标函数，该数学描述的含义亦即求取一组 x 向量，使得最优化目标函数 $f(x)$ 为最小，故这样的问题又称为最小化问题。其实，最小化是最优化问题的通用描述，它不失普遍性。如果要想求解最大化问题，那么只需给目标函数 $f(x)$ 乘以一个负号就能立即将最大化问题转化为最小化问题。所以本书中描述的所有问题都是最小化问题。

1．解析解法和图解法

无约束最优化问题的最优点 x^* 处，目标函数 $f(x)$ 对 x 的各个分量的一阶导数为 0，从而可以列出下面的方程：

$$\frac{\partial f}{\partial x_1}\Big|_{x=x^*}=0,\frac{\partial f}{\partial x_2}\Big|_{x=x^*}=0,\cdots,\frac{\partial f}{\partial x_n}\Big|_{x=x^*}=0$$

求解这些方程构成的联立方程可以得出极值点。其实，解出的一阶导数均为 0 的极值点不一定都是极小值的点，其中有的还可能是极大值点。极小值问题还应该有正的二阶导数。对于单变量的最优化问题，可以考虑采用解析的方法进行求解。然而多变量最优化问题因为需要将其转换成求解多元非线性方程，其难度也不低于直接求取最优化问题，所以没有必要采用解析方法求解。

一元函数最优化问题的图解法也是很直观的，应绘制出该函数的曲线，在曲线上就能看出其最优值点。二元函数的最优化也可以通过图解法求出。但三元或多元函数，由于用图形没有办法表示，所以不适合用图解法求解。

例 5.33 已知一元函数 $f(t)=e^{-3t}\sin(4t+2)+4e^{-0.5t}\cos(2t)-0.5$，$t\in[0,4]$。试用解析求解和图形求解的方法研究该函数的最优性。

解：可以先表示该函数，并解析求解该函数的一阶导数，用 ezplot()函数可以绘制出

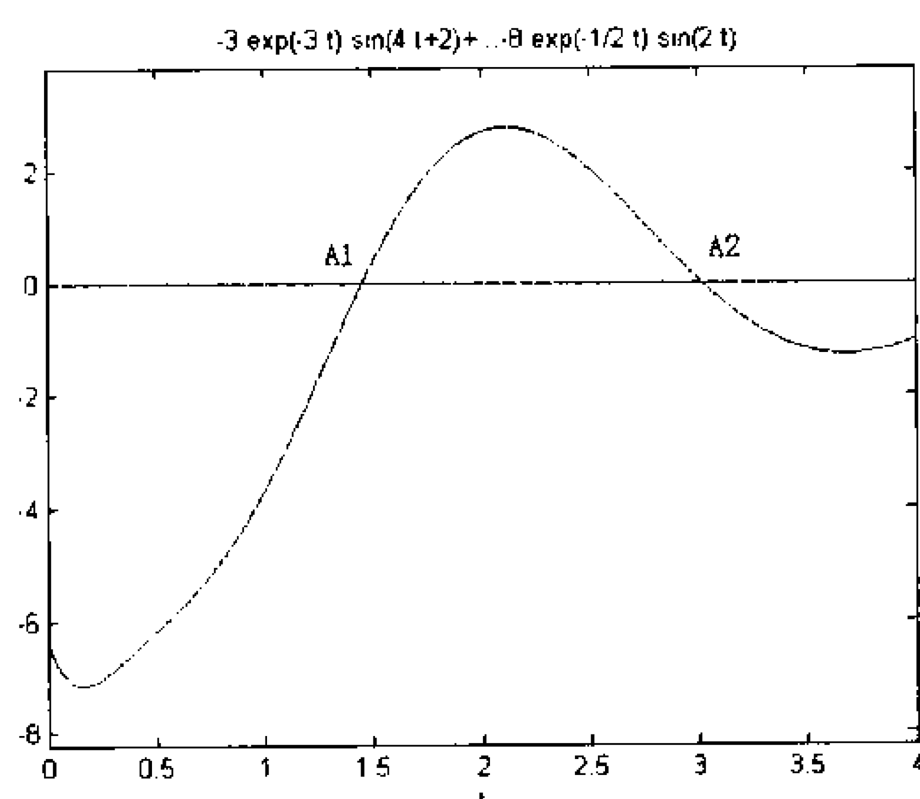
$t \in [0,4]$ 区间内一阶导函数的曲线，如图 5-25（a）所示。

```
>> syms t;y=exp(-3*t)*sin(4*t+2)+4*exp(-0.5*t)*cos(2*t)-0.5;
>> y1=diff(y,t); %求取一阶导函数
>> ezplot(t-t,[0,4])
>> hold on
>> ezplot(y1,[0,4]) %绘制出选定区间内一阶导函数曲线
```

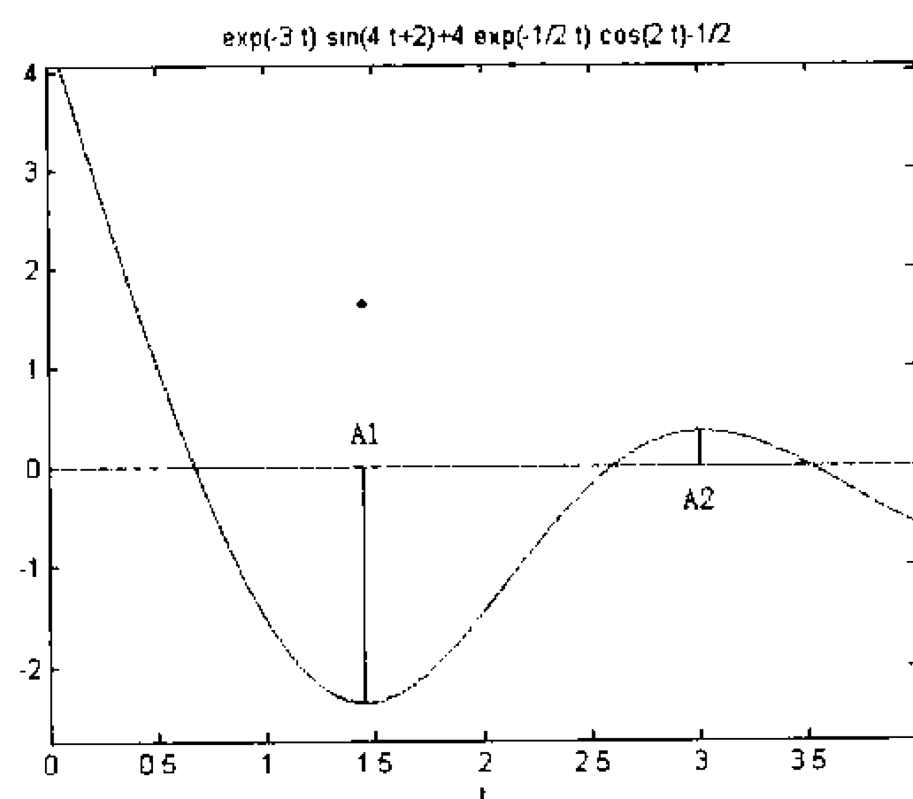
其实，求解导函数等于 0 的方程不比直接求解其最优值简单。用图解法可以看出，在这个区间内有两个点，A_1 和 A_2，使得 $f(t)$ 的一阶导函数为 0，但从其一阶导函数走向看，A_2 点对应负的二阶导数值，所以该点对应于极大值点，而 A_1 点对应正的二阶导数值，故为极小值点。A_1 点的值可由下面的语句直接求出。

```
>> t0=solve(y1) %求出一阶导数等于 0 的点
t0 =
1.4528424981725411893375778048840
>> ezplot x-x
>> hold on，ezplot(y,[0,4])
>> y2=diff(y1);b=subs(y2,t,t0) %并验证二阶导数为正
b =
7.8553420253333601379464405534590
```

这样，就可以求出函数的极小值。还可以用图形绘制的方法进一步验证得出的结果，如图 5-25（b）所示，可见，A_1 为最小值点，A_2 为最大值点。



（a）函数的一阶导函数曲线



（b）函数曲线

图 5-25　联立方程图解法示意图

但是，因为给定的函数是非线性函数，所以用解析法或类似的方法求解最小值问题一点都不比直接求解最优化问题简单。所以，除演示外，不建议用这样的方法求解该问题，而直接采用最优化问题求解程序得出问题的解。

2. 基于 MATLAB 的数值解法

MATLAB 语言中提供了 3 个求解无约束最优化问题的函数：fminbnd()、fminsearch() 和 fminunc()。其调用格式如下：

- [x,fval,exitflag,output]=fminbnd('fun',x1,x2,options,p1,p2,...)
- [x,fval,exitflag,output]=fminsearch('fun',x0,options,p1,p2,...)
- [x,fval,exitflag,output,grad,hessian]=fminunc('fun',x0,options,p1,p2,...)

对其中各参数说明如下：

①fminbnd 用于求单变量函数在区间（x1,x2）上的极小值点，'fun'，x1 和 x2 是不可默认的输入参数。fminsearch 和 fminunc 用于求多变量函数的极小值点，'fun'和'x0'是不可默认的输入参数。'fun'为指定的目标函数，可以通过建立 m-文件（function）来定义，也可有 inline 命令在 MATLAB 指令窗口中给出。x0 是极小值点的初始近似值。x 是不可默认的输出参数，它是问题的解。

②options 是用来控制算法的参数，通过 optimset 函数对其进行设置。3 个函数可以选用的控制算法选项如表 5-1 所示。

表 5-1　各函数可选择的算法选项

函　数	算 法 选 项
fminbnd	Display,TolX,MaxFunEval,MaxIter,FunValCheck, PlotFcns, OutputFcn
fminsearch	Display, TolX, TolFun, MaxFunEvals, MaxIter, FunValCheck,PlotFcns,OutputFcn
fminunc	Display,TolX,TolFun,DerivativeCheck,Diagnostics, FunValCheck,GradObj,HessPattern,Hessian,HessMult, HessUpdate,InitialHessType, InitialHessMatrix, MaxFunEvals,MaxIter,DiffMinChange,DiffMaxChange, LargeScale,MaxPCGIter,PrecondBandWidth,TolPCG, PlotFcns, OutputFcn, TypicalX

如果使用选项参数 GradObj，在目标函数的 m-文件中应有两项输出，其第二项输出为目标函数的梯度向量。如果使用选项参数 Hessian，在目标函数的 m-文件中应有三项输出，其第二项输出为目标函数的梯度向量，第三项输出为目标函数的二阶偏导数矩阵（Hessian 矩阵）。选项参数 Hessian 仅在选用 Large-Scale 算法时使用，在选用 Line-search 算法时不使用。

③参数 p1,p2,…是向目标函数传递的参数的值。

④输出参数 fval 是目标函数在解 x 处的值。

⑤输出参数 exitflag 的值描述了程序的运行情况。对于程序 fminunc，如果 exitflag 的值大于 0，则程序收敛于解 x；如果 exitflag 的值等于 0，则函数的计算达到了最大次数；如果 exitflag 的值小于 0，则程序未收敛到解。对于程序 fminbnd 和 fminsearch，如果 exitflag 的值等于 1，则程序收敛于解 x；如果 exitflag 的值等于 0，则达到了最大迭代次数。

⑥输出参数 output 的值为算法的迭代次数。

⑦输出参数 grad 为目标函数在解 x 处的梯度。

⑧输出参数 hessian 为目标函数在解 x 处的 Hessian 矩阵。

例 5.34　已知函数 $f(x,a)=\dfrac{x}{a+x^2}$。当 a=1 时，求函数在区间[-2,1]内的极小值点及极小值。

解：第一步：建立被求极小值函数的 m-文件。在文件编辑窗口输入：

```
function y=e1411(x,a)
y=x./(a+x.^2);
```

第二步：求解。由函数可知，该题目是一元函数求极小值，故选用 fminbnd()函数。在指令窗口输入：

```
>> x1=-2;x2=1;
```

```
>> options=optimset('tolx',1e-004);
>> [x,f]=fminbnd(@e1411,x1,x2,options,1)
```

运行后得到输出：

```
x =
    -1.0000
f =
    -0.5000
```

例 5.35 已知二元函数 $f(x)=e^{x_1}(4x_1^2+2x_2^2+4x_1x_2+2x_2+1)$，求其极小值。

解：第一步：建立目标函数的 m-文件：

```
function y=e1412(x)
y=exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
```

第二步：求解。在指令窗口输入：

```
>> x0=[-1,1];
>> [x,y]=fminunc('e1412',x0)
```

运行后得出极小值点和目标函数的极小值：

```
x =
    0.5000    -1.0000
y =
    3.6609e-015
```

如果在求极小值时使用函数的梯度，则在目标函数的 m-文件中应有两个输出。第二个输出为目标函数的梯度向量：

```
function [y,g]=e1412b(x)
y=exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
g=[exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1+8*x(1)+4*x(2)),...
    exp(x(1))*(4*x(2)+2+4*x(1))];
```

在指令窗口输入：

```
>> x0=[-1,1];
>> options=optimset('gradobj','on');
>> [x,y]=fminunc(@e1412b,x0,options)
```

运行后得出极小值点和目标函数的极小值：

```
x =
    0.5000    -1.0000
y =
  1.3911e-014
```

在求解时，也可不建立 m-文件，而在指令窗口输入：

```
>> f=inline('exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1)');
>> x0=[-1,1];
>> [x,y]=fminunc(f,x0)
```

运行后得到目标函数极小值点和目标函数的极小值：

```
x =
    0.5000    -1.0000
y =
    3.6609e-015
```

5.8.2 有约束最优化问题求解

有约束最优化问题的一般描述为：

$$\min_{x \text{ s.t. } G(x)\leqslant 0} f(x)$$

其中，$x=[x_1,x_2,\cdots,x_n]^T$，记号 s.t.是英文 subject to 的缩写，表示满足后面的关系。该数学表达式表示的含义为求取一组 x 向量，使得在满足约束条件 $G(x)\leqslant 0$ 的前提下能够使目标函数 $f(x)$ 最小化。在实际遇到的最优化问题中，有时约束条件可能是很复杂的，它既可以是等式约束，也可以是不等式约束，可以是线性的，也可以是非线性的，有时甚至不能用纯数学函数来描述。

在 MATLAB 的优化工具箱中有一个求解有约束最优化问题的函数 fmincon()。其调用格式如下：

[x,fval,exitflag,output,lambda,grad,hession]=fmincon('fun',x0,A,b,A1,b1,LB,UB,'nonlcon',options,p1,p2,...)

现对其中参数说明如下：

①fmincon()函数用于求解多变量函数有约束最优化问题：

$$\min_{x \text{ s.t.} \begin{cases} Ax\leqslant b, \\ A_1x=b_1, \\ C(x)\leqslant 0, \\ C_1(x)=0, \\ LB\leqslant x\leqslant UB. \end{cases}} f(x)$$

其中 $Ax\leqslant b$ 和 $A_1x=b_1$ 是线性约束，$C(x)\leqslant 0$ 和 $C_1(x)=0$ 是非线性约束，$LB\leqslant x\leqslant UB$ 是有界约束。

②'fun'和 x0 是不可默认的输入参数。fun 是给出目标函数的 m-文件的名字，目标函数也可由 inline 函数在 MATLAB 的指令窗口给出。x0 是极小值点的初始近似值。x 是不可默认的输出参数，它是问题的解。

③nonlcon 是给出非线性约束函数 $C(x)$ 和 $C_1(x)$ 的 m-文件的文件名。文件的输出为 $[C(x), C_1(x)]$。

④当 x 无下界时，在 LB 处放置[]。当无上界时，在 UB 处放置[]。如果 x 的某个分量 x_j 无下界，则置 LB(j)=-inf。如果 x_j 无上界，则置 UB(j)=inf。如果无线性不等式约束，则在 A 和 b 处都放置[]。

⑤options 是用来控制算法的选项参数。fmincon()函数可以选择的 options 选项可以通过 help 进行查看。在使用 GradObj 选项时，fun 应有两项输出[f,G]，第二项输出 G 是函数 $f(x)$的偏导数。在使用 GradConstr 时，nonlcon 中应有四项输出 $[C, C_1, GC, GC_1]$，其中 GC 是不等式约束函数 C(x)的偏导数，GC1 是等式约束函数 $C_1(x)$的偏导数。

⑥参数 p1,p2,…是向目标函数传送的参数的值。

⑦输出参数 fval 是目标函数在解 x 处的值。

⑧输出参数 exitflag 的值描述了程序的运行情况。如果 exitflag 的值大于 0，则程序收敛于解 x；如果 exitflag 的值等于 0，则函数的计算达到了最大次数；如果 exitflag 的值小于

0，则程序未收敛到解。

⑨输出参数 output 输出程序运行的某些信息。其中 output.iterations 是迭代次数，output.funccount 是函数的计算次数，output.algorithm 是所使用的算法，output. Cgiterations 是共轭梯度（如果使用了）迭代次数，output. Firstorderopt 是一阶最优性（如果使用了）的度量。

⑩输出参数 grad 为目标函数在解 x 处的梯度。

⑪输出参数 Lambda 为在解 x 处的 Lagrange 乘子。其中 Lambda.lower 是对应于 LB 的，Lambda.eqlin 是对应于线性等式约束的，Lambda.ineqnonlin 是对应于非线性不等式约束的，Lambda.inequalities 是对应于非线性等式约束的。

⑫输出参数 hession 为目标函数在解 x 处的 Hession 矩阵。

例 5.36 求解最优化问题：

$$\min_{x \text{ s.t.} \begin{cases} x_1+x_2=0 \\ 1.5+x_1x_2-x_1-x_2\leqslant 0 \\ -x_1x_2-10\leqslant 0 \end{cases}} e^{x_1}\left(4x_1^2+2x_2^2+4x_1x_2+2x_2+1\right)$$

解：第一步：建立目标函数和非线性约束函数的 m-文件：

```
function y=e1511(x)
y=exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
function [c1,c2]=e1511b(x)
c1=[1.5+x(1)*x(2)-x(1)-x(2);-x(1)*x(2)-10];
c2=[];
```

第二步：调用 fmincon()函数进行求解，在指令窗口输入：

```
>> x0=[-1,1];a1=[1,1];b1=0;
>> [x,f,exitflag,output]=fmincon(@e1511,x0,[],[],a1,b1,[],[],@e1511b)
```

运行后得到输出：

```
x =
    -1.2247      1.2247
f =
    1.8951
exitflag =
     1
output =
          iterations: 4
           funcCount: 15
       lssteplength: 1
            stepsize: 3.7569e-008
           algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
      firstorderopt: 2.2204e-016
             message: [1x143 char]
```

由输出结果可以看出，经过 4 次迭代（iterations: 4）收敛到了（exitflag =1）最优解 x(1)=–1.2247,x(2)=1.2247，目标函数最优值为 1.8951。

例 5.37 试求解下面的有约束最优化问题：

$$\min_{x \text{ s.t.} \begin{cases} x_1^2+x_2^2+x_3^2-25=0 \\ 8x_1+14x_2+7x_3-56=0 \\ x_1,\ x_2,\ x_3\geqslant 0 \end{cases}} \left[1000-x_1^2-2x_2^2-x_3^2-x_1x_2-x_1x_3\right]$$

解：根据给出的问题，可以直接写出目标函数为：

```
function y=opt_fun1(x)
y=1000-x(1)^2-2*x(2)^2-x(3)^2-x(1)*x(2)-x(1)*x(3);
```

同时，给出的两个约束条件有一个为非线性约束，所以应该写出非线性约束函数为：

```
function [c1,c2]=opt_con1(x)
c1=[];
c2=x(1)^2+x(2)^2+x(3)^2-25;
```

非线性的约束函数返回变量分为 c1 和 c2 两个量，其中，前者为非线性不等式约束的数学描述，后者为非线性等式约束的数学描述，如果某个约束不存在，则应将其值赋为空矩阵。这样的约束函数处理比早期版本的工具箱中处理更方便、规范。

描述了给出的非线性等式约束后，则 ***A***、***b*** 为空矩阵，A1=[8,14,7]，b1=56，LB=[0;0;0]，UB=[inf;inf;inf]。此外，应该给出初始近似值 x0=[1;1;1]。描述完所有约束条件后，就可以调用 fmincon()函数求解此约束最优化问题。

```
>> ff=optimset;ff.LargeScale='off';ff.Display='iter';
>> ff.TolFun=1e-30;ff.TolX=1e-15;ff.TolCon=1e-20;
>> x0=[1;1;1];A1=[8,14,7];b1=56;LB=[0;0;0];UB=[inf;inf;inf];
>> [x,f,exitflag,output]=fmincon(@opt_fun1,x0,[],[],A1,b1,LB,UB,@opt_con1,ff);
```

运行后得到：

```
                         max      Line search  Directional  First-order
 Iter F-count     f(x)   constraint  steplength   derivative   optimality  Procedure
 0    4       994        27                                              Infeasible start point
 1    10      955.336    22.9         0.25        -295         18.3      infeasible
 2    14      964.012    5.773        1           0.811        6.26      Hessian modified; dependent
 3    24      965.164    5.752        0.0156      70.1         1.21e+004 Hessian modified; infeasible
 4    28      957.066    5.298        1           -11.3        441
 5    34      956.19     4.749        0.25        -4.85        394       infeasible
 6    38      961.043    0.5622       1           4.33         5.58e+003 Hessian modified
 7    42      961.759    0.1325       1           0.658        3.23e+004 Hessian modified
 8    46      961.918    0.0003214    1           0.158        622       Hessian modified
 9    50      961.918    4.953e-009   1           0.000441     3.02
 10   54      961.918    7.105e-015   1           2.36e-009    0.394     Hessian modified twice
 11   58      961.918    1.645e-012   1           -1.04e-006   1.16e+004 Hessian modified twice
 12   62      961.918    7.105e-015   1           5.99e-008    108       Hessian modified
 13   66      961.918    0            1           6.52e-015    2.59      Hessian modified twice
Optimization terminated: magnitude of search direction less than 2*options.TolX
 and maximum constraint violation is less than options.TolCon.
No active inequalities.
x =
    3.1311
    0.2662
    3.8891
f =
    961.9182
exitflag =
     4
output =
       iterations: 13
        funcCount: 66
     lssteplength: 1
```

```
        stepsize: 1.4016e-015
       algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
   firstorderopt: 2.9802e-008
         message: [1x142 char]
```

由输出结果可以看出，经过 13 次迭代（iterations: 13）收敛到了（exitflag =4）最优解 x(1)= 3.1311，x(2)= 0.2662，x(3)=3.8891，目标函数最优值为 961.9182。

5.8.3 基于遗传算法的最优化

作为一种新的全局优化搜索算法，遗传算法以其简单通用、鲁棒性强、适于并行处理以及高效实用等显著特点，在各个领域得到了广泛应用，取得了良好效果。

1. 遗传算法简介

遗传算法（Genetic Algorithm，缩写为 GA）是一种有效地解决最优化问题的方法。它最先是由 John Holland 于 1975 年提出的。遗传算法是模拟达尔文的遗传选择和自然淘汰的生物进化过程的计算模型。它的思想源于生物遗传学和适者生存的自然规律，是具有“生存＋检测”的迭代过程的搜索算法。遗传算法以一种群体中的所有个体为对象，并利用随机化技术指导对一个被编码的参数空间进行高效搜索。其中，选择、交叉和变异构成了遗传算法的遗传操作；参数编码、初始群体的设定、适应度函数的设计、遗传操作设计和控制参数设定等 5 个要素组成了遗传算法的核心内容。

2. 遗传算法的基本步骤

遗传算法是一种基于生物自然选择与遗传机理的随机搜索算法，与传统搜索算法不同，遗传算法从一组随机产生的初始解，称为“种群（Population）”开始搜索过程。种群中的每个个体式问题的一个解，称为“染色体（Chromosome）”。染色体是一串符号，比如一个二进制字符串。这些染色体在后续迭代中不断进化，称为遗传。在每一代中用“适值（Fitness）”来测量染色体的好坏，生成的下一代染色体称为后代（Offspring）。后代是由前一代染色体通过交叉（Crossover）或者变异（Mutation）运算形成的。在新一代形成中，根据适度的大小选择部分后代，淘汰部分后代，从而保持种群大小是常数。适值高的染色体被选中的概率较高。这样经过若干代之后，算法收敛于最好的染色体，它很可能就是问题的最优解或次优解。

遗传算法的主要步骤如下所示。

（1）编码：GA 在进行搜索之前先将解空间的解数据表示成遗传空间的基因型串结构数据，这些串结构数据的不同组合便构成了不同的点。

（2）初始群体的生成：随机产生 N 个初始串结构数据，每个串结构数据称为一个个体，N 个个体构成了一个群体。GA 以这 N 个串结构数据作为初始点开始迭代。

（3）适应性值评估检测：适应性函数表明个体或解的优劣性。对于不同的问题，适应性函数的定义方式也不同。

（4）选择：选择的目的是为了从当前群体中选出优良的个体，使它们有机会作为父代为下一代繁殖子孙。遗传算法通过选择过程体现这一思想，进行选择的原则是适应性强的个体为下一代贡献一个或多个后代的概率大。选择实现了达尔文的适者生存原则。

（5）交叉：交叉操作是遗传算法中最主要的遗传操作。通过交叉操作可以得到新一代个体，新个体组合了其父辈个体的特性。交叉体现了信息交换的思想。

（6）变异：变异首先在群体中随机选择一个个体，对于选中的个体以一定的概率随机地改变串结构数据中某个串的值。同生物界一样，GA 中变异发生的概率很低，通常取值在 0.001~0.01 之间。变异为新个体的产生提供了机会。

实际上，遗传算法中有以下两类运算。

- 遗传运算：交叉和变异。
- 进化运算：选择。

遗传算法模拟了基因在每一代中创造新后代的繁殖过程，进化运算则是种群逐代更新的过程。交叉是最主要的遗传运算，它同时对两个染色体操作，组合二者的特性产生新的后代。交叉的最简单方法是在双亲（两个父辈的个体）的染色体上随机地选一个断点，将断点的右段互相交换，从而形成两个新的后代。这种方法对于二进制表示最为合适。遗传算法的性能在很大程度上取决于采用的交叉运算的性能。变异则是一种基本运算，它在染色体上自发地产生随机的变化。一种简单的变异方法是替换一个或多个基因。在遗传算法中，变异可以提供初始种群中不含有的基因，或找回选择过程丢失的基因，为种群提供新的内容。

GA 的计算过程流程图如图 5-26 所示。

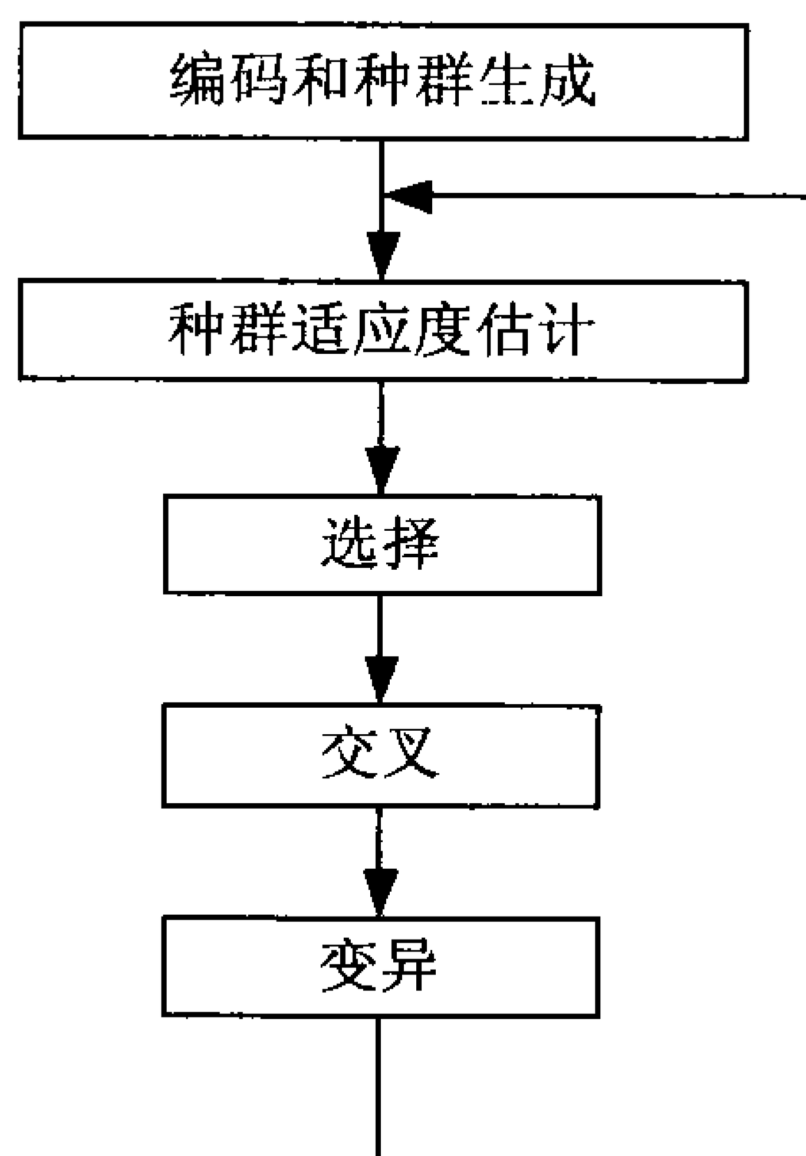


图 5-26　GA 的计算过程流程图

3．遗传算法的特点

GA 算法具有下述几个特点。

- GA 是对问题参数的编码组进行进货，而不是直接针对参数本身。
- GA 的搜索是从问题解的编码组开始搜索，而不是从单个解开始。
- GA 使用目标函数值（适应度）这一信息进行搜索，而不需导数等其他信息。
- GA 算法使用的选择、交叉和变异这 3 个算子都是随机操作，而不是确定规则。

4．遗传算法在工程优化中的应用

最优化问题通常可归结为极大化问题，利用数字公式描述可以写为：

$$\max_{x \in S} f(X)$$

其中$f(x)$为目标函数，S为可行域，它们是由工程实际问题的具体条件决定的。

实践表明，遗传算法解最优化问题的计算效率比较高，适用范围相当广。为了解释这一现象，Holland 给出了图式定理。所谓图式，就是某些码位取相同值的编码的集合。图式定理说明在进化过程的各代中，属于适应度高、阶数低且长度短的图式的编码数量将随代数以指数形式增长。另外，Holland 还发现遗传算法具有隐含的并行计算特性。最近的研究则表明，上述遗传算法经适当改进后对任意优化问题以概率 1 收敛于全局最优解。

利用遗传算法解最优化问题，首先应对可行域中的点进行编码(一般采用二进制编码)，然后在可行域中随机挑选一些编码组成作为进化起点的第一代编码组，并计算每个解的目标函数值，也就是编码的适应度。接着就像自然界中一样，利用选择机制从编码组中随机挑选编码作为繁殖过程前的编码样本。选择机制应保证适应度较高的解能够保留较多的样本；而适应度较低的解则保留较少的样本，甚至被淘汰。在接下去的繁殖过程中，遗传算法提供了交叉和变异两种算子对挑选后的样本进行交换。交叉算子交换随机挑选的两个编码的某些位，变异算子则直接对一个编码中的随机挑选的某一位进行反转。这样通过选择和繁殖就产生了下一代编码组。重复上述选择和繁殖过程，直到结束条件得到满足为止。进化过程最后一代中的最优解就是用遗传算法解最优化问题所得到的最终结果。

5. 遗传算法的 MATLAB 实现

前面对遗传算法及其功能进行了简介，那么怎样用 MATLAB 来实现呢？本节将对具体的实现方法进行详细阐述，并穿插了很多例程。

1）编码和种群生成

通过参数 options 来选择编码的形式：浮点编码或二进制编码。

参数说明：

pop：生成的初始种群。

populatoinSize：种群中的个体的数目。

variableBounds：表示变量边界的矩阵。

eevalFN：适应度函数。

eevalOps：传递给适应度函数的参数。

以下是求解的 MATLAB 代码：

```
function [pop] = initializega(num, bounds, eevalFN,eevalOps,options)
if nargin<5
   options=[1e-6 1];
end
if nargin<4
   eevalOps=[];
end
if options(2)==1   %浮点编码
      estr=['[ pop(i,:) pop(i,xZomeLength)]=' eevalFN '(pop(i,:),[0 eevalOps]);'];
   else   %二进制编码
      estr=['x=b2f(pop(i,:),bounds,bits);[x v]=' eevalFN(x,[0 eevalOps]);
pop(i,:)=[f2b(x,bounds,bits) v];'];
   end
```

```
end

numVars = size(bounds,1);        %变量的数目
rng= (bounds(:,2)-bounds(:,1))';   %变量边界

if options(2)==1
   xZomeLength = numVars+1;
   pop= zeros(num,xZomeLength);
   pop(:,1:numVars)=(ones(num,1)*rng).*(rand(num,numVars))+ (ones(num,1)*bounds(:,1)');
else
   bits=calcbits(bounds,options(1));
   xZomeLength = sum(bits)+1;
   pop = round(rand(num,sum(bits)+1));
end

for i=1:num
   eeval(estr);
end
```

2）交叉

交叉过程选择两个个体作为父代，产生两个新的子代个体。

参数说明：

p1：第一个父代个体。

p2：第二个父代个体。

bounds：可行域的边界矩阵。

以下是求解的 MATLAB 代码：

```
function [c1,c2]=crossover(p1,p2,bounds,ops)
sz=size(p1,2)-1;
right=[2:sz 1];
left =[sz 1:(sz-1)];
p1i(p1)=1:sz; %生成序列号
p2i(p2)=1:sz; %生成序列号
adj=sort([-1:-1:-sz;p1(right(p1i));p1(left(p1i));p2(right(p2i));p2(left(p2i))]');
repeats=find(diff(adj(:,2:5)')'==0);
adj(repeats+sz)=zeros(size(repeats));

curr_site = round(rand*sz + 0.5);选择任意交叉点
for site=1:(sz-1)
    c1(site)=curr_site;
    inAdj=find(adj(:,2:5)==curr_site);
    adj(inAdj+sz)=zeros(size(inAdj));
    lz=colperm(adj');
    lzi(lz)=1:size(lz,2); %create index
    adj_cities=adj(curr_site,1+(find(adj(curr_site,2:5))));
    [v i]=min(lzi(adj_cities));
    curr_site=adj_cities(i);
end
c1(sz)=curr_site;
c2=p1;
```

3）变异

变异操作由一个父代产生单个子代。

以下是求解的 MATLAB 代码：

```
function [child] = Mutation(par,bounds,genInfo,Ops)
sz = size(par,2)-1;
ppos = round(rand*sz + 0.5);
cpos = round(rand*sz + 0.5);
if ppos ~= cpos
    if ppos > cpos
        child = [par(1:cpos-1) par(ppos) par(cpos:ppos-1) par(ppos+1:sz) par(sz+1)];
    else
        child = [par(1:ppos-1) par(ppos+1:cpos) par(ppos) par(cpos+1:sz) par(sz+1)];
    end
else
    child = par;
end
```

4）选择

选择操作决定哪些个体可以进入下一代，程序中采用标准几何分布进行选择。

参数说明：

newPop：由父代种群选出新的种群。

oldPop：当前的种群（父代种群）。

以下是求解的 MATLAB 代码：

```
function[newPop] = Select(oldPop,options)
q=options(2);                       % 选择最优的概率
e = size(oldPop,2);
n = size(oldPop,1);                 % 种群中个体的数目
newPop = zeros(n,e);                % 初始化
fit = zeros(n,1);                   % 初始化
x=zeros(n,2);
x(:,1) =[n:-1:1]';
 [y x(:,2)] = sort(oldPop(:,e));
r = q/(1-(1-q)^n);                  % 标准分布
fit(x(:,2))=r*(1-q).^(x(:,1)-1);    % 生成选择概率
fit = cumsum(fit);                  % 计算选择概率之和
rNums=sort(rand(n,1));
fitIn=1; newIn=1;
while newIn<=n                      % 得到新的个体
   if(rNums(newIn)<fit(fitIn))
      newPop(newIn,:) = oldPop(fitIn,:); % 选择合适的个体
      newIn = newIn+1;
   else
      fitIn = fitIn + 1;
   end
end
```

以下是完整的遗传算法函数程序：

```
function [x,endPop,bPop,traceInfo] = ga(bounds,eevalFN,eevalOps,startPop,opts,...
termFN,termOps,selectFN,selectOps,xOverFNs,xOverOps,mutFNs,mutOps)
```

```
n=nargin;
if n<2 | n==6 | n==10 | n==12
  disp('Insufficient arguements')
end
if n<3 %Default eevalation opts.
  eevalOps=[];
end
if n<5
  opts = [1e-6 1 0];
end
if isempty(opts)
  opts = [1e-6 1 0];
end

if any(eevalFN<48) %Not using a .m file
  if opts(2)==1 %Float ga
    e1str=['x=c1; c1(xZomeLength)=', eevalFN ';'];
    e2str=['x=c2; c2(xZomeLength)=', eevalFN ';'];
  else %Binary ga
    e1str=['x=b2f(endPop(j,:),bounds,bits); endPop(j,xZomeLength)=',...
eevalFN ';'];
  end
else %Are using a .m file
  if opts(2)==1 %Float ga
    e1str=['[c1 c1(xZomeLength)]=' eevalFN '(c1,[gen eevalOps]);'];
    e2str=['[c2 c2(xZomeLength)]=' eevalFN '(c2,[gen eevalOps]);'];
  else %Binary ga
    e1str=['x=b2f(endPop(j,:),bounds,bits);[x v]=' eevalFN ...
'(x,[gen eevalOps]); endPop(j,:)=[f2b(x,bounds,bits) v];'];
  end
end


if n<6 %Default termination information
  termOps=[100];
  termFN='maxGenTerm';
end
if n<12 %Default muatation information
  if opts(2)==1 %Float GA
  mutFNs=['boundaryMutation multiNonUnifMutation nonUnifMutation unifMutation'];
    mutOps=[4 0 0;6 termOps(1) 3;4 termOps(1) 3;4 0 0];
  else %Binary GA
    mutFNs=['binaryMutation'];
    mutOps=[0.05];
  end
end
if n<10      %默认的交叉信息
  if opts(2)==1      %浮点编码
    xOverFNs=['arithXover heuristicXover simpleXover'];
```

```
            xOverOps=[2 0;2 3;2 0];
        else %Binary GA
            xOverFNs=['simpleXover'];
            xOverOps=[0.6];
        end
    end
    if n<9 %Default select opts only i.e. roullete wheel.
        selectOps=[];
    end
    if n<8      %Default select info
        selectFN=['normGeomSelect'];
        selectOps=[0.08];
    end
    if n<6      %默认的算法终止准则
        termOps=[100];
        termFN='maxGenTerm';
    end
    if n<4      %初始种群为空
        startPop=[];
    end
    if isempty(startPop)      %随机生成初始种群
        startPop=initializega(80,bounds,eevalFN,eevalOps,opts(1:2));
    end

    if opts(2)==0      %二进制编码
        bits=calcbits(bounds,opts(1));
    end

    xOverFNs=parse(xOverFNs);
    mutFNs=parse(mutFNs);

    xZomeLength    = size(startPop,2);                 %Length of the xzome=numVars+fittness
    numVar         = xZomeLength-1;                    %变量数目
    popSize        = size(startPop,1);                 %种群中个体数目
    endPop         = zeros(popSize,xZomeLength);       %次种群矩阵
    c1             = zeros(1,xZomeLength);             %个体
    c2             = zeros(1,xZomeLength);             %个体
    numXOvers      = size(xOverFNs,1);                 %交叉操作次数
    numMuts        = size(mutFNs,1);                   %变异操作次数
    epsilon        = opts(1);                          %适应度门限值
    oeval          = max(startPop(:,xZomeLength)); %初始种群中的最优值
    bFoundIn       = 1;
    done           = 0;
    gen            = 1;
    collectTrace = (nargout>3);
    floatGA        = opts(2)==1;
    display        = opts(3);

    while(~done)
        [beval,bindx] = max(startPop(:,xZomeLength));       %当前种群的最优值
```

```
    best =   startPop(bindx,:);

    if collectTrace
      traceInfo(gen,1)=gen;                          %当前代
      traceInfo(gen,2)=startPop(bindx,xZomeLength);            %最优适应度
      traceInfo(gen,3)=mean(startPop(:,xZomeLength));        %平均适应度
      traceInfo(gen,4)=std(startPop(:,xZomeLength));
    end

    if ( (abs(beval - oeval)>epsilon) | (gen==1))
      if display
        fprintf(1,'\n%d %f\n',gen,beval);
      end
      if floatGA
        bPop(bFoundIn,:)=[gen startPop(bindx,:)];
      else
        bPop(bFoundIn,:)=[gen b2f(startPop(bindx,1:numVar),bounds,bits)...
         startPop(bindx,xZomeLength)];
      end
      bFoundIn=bFoundIn+1;
      oeval=beval;
    else
      if display
        fprintf(1,'%d ',gen);
      end
    end

    endPop = feeval(selectFN,startPop,[gen selectOps]);       %选择操作

    if floatGA
      for i=1:numXOvers,
        for j=1:xOverOps(i,1),
      a = round(rand*(popSize-1)+1);       %一个父代个体
      b = round(rand*(popSize-1)+1);       %另一个父代个体
      xN=deblank(xOverFNs(i,:));          %交叉函数
      [c1 c2] = feeval(xN,endPop(a,:),endPop(b,:),bounds,[gen... xOverOps(i,:)]);

      if c1(1:numVar)==endPop(a,(1:numVar))
        c1(xZomeLength)=endPop(a,xZomeLength);
      elseif c1(1:numVar)==endPop(b,(1:numVar))
        c1(xZomeLength)=endPop(b,xZomeLength);
      else
        eeval(e1str);
      end
      if c2(1:numVar)==endPop(a,(1:numVar))
        c2(xZomeLength)=endPop(a,xZomeLength);
      elseif c2(1:numVar)==endPop(b,(1:numVar))
        c2(xZomeLength)=endPop(b,xZomeLength);
      else
        eeval(e2str);
```

```
        end

        endPop(a,:)=c1;
        endPop(b,:)=c2;
                end
            end

      for i=1:numMuts,
        for j=1:mutOps(i,1),
        a = round(rand*(popSize-1)+1);
        c1 = feeval(deblank(mutFNs(i,:)),endPop(a,:),bounds,[gen mutOps(i,:)]);
        if c1(1:numVar)==endPop(a,(1:numVar))
          c1(xZomeLength)=endPop(a,xZomeLength);
        else
          eeval(e1str);
        end
        endPop(a,:)=c1;
              end
          end

      else      %遗传操作的统计模型
        for i=1:numXOvers,
              xN=deblank(xOverFNs(i,:));
              cp=find(rand(popSize,1)<xOverOps(i,1)==1);
              if rem(size(cp,1),2) cp=cp(1:(size(cp,1)-1)); end
              cp=reshape(cp,size(cp,1)/2,2);
              for j=1:size(cp,1)
        a=cp(j,1); b=cp(j,2);
        [endPop(a,:) endPop(b,:)] = feeval(xN,endPop(a,:),endPop(b,:), bounds,[gen xOverOps(i,:)]);
                end
            end
      for i=1:numMuts
        mN=deblank(mutFNs(i,:));
        for j=1:popSize
      endPop(j,:) = feeval(mN,endPop(j,:),bounds,[gen mutOps(i,:)]);
      eeval(e1str);
        end
      end
    end

    gen=gen+1;
    done=feeval(termFN,[gen termOps],bPop,endPop); %
    startPop=endPop;                  %更新种群

    [beval,bindx] = min(startPop(:,xZomeLength));
    startPop(bindx,:) = best;
  end

  [beval,bindx] = max(startPop(:,xZomeLength));
  if display
```

```
    fprintf(1,'\n%d %f\n',gen,beval);
  end

  x=startPop(bindx,:);
  if opts(2)==0 %binary
      x=b2f(x,bounds,bits);
  bPop(bFoundIn,:)=[gen b2f(startPop(bindx,1:numVar),bounds,bits), startPop(bindx,xZomeLength)];
  else
    bPop(bFoundIn,:)=[gen startPop(bindx,:)];
  end
  if collectTrace
    traceInfo(gen,1)=gen;
    traceInfo(gen,2)=startPop(bindx,xZomeLength); %Best fittness
    traceInfo(gen,3)=mean(startPop(:,xZomeLength)); %Avg fittness
  end
```

参数说明：

①输出参数：

x：求得最优解。

endPop：最终的种群。

bPop：最优种群的一个搜索轨迹。

②输入参数：

bounds：代表变量的上下界的矩阵。

eevalFN：适应度函数。

startPop：初始群体。

termFN：终止函数的名称。

termOps：终止函数的参数。

selectFN：选择函数的名称。

selectOpts：选择参数。

6．范例分析

利用遗传算法计算下列函数的最大值：

$$f(x) = x + 10 * \sin(5x) + 7 * \cos(4x)$$
$$x \in [0,9]$$

解：选择二进制编码，种群中的个体数目为 10，二进制编码序列的长度为 20，交叉概率为 0.95，变异概率为 0.08。

例程 5.6 是求解的 MATLAB 代码。

例程 5.6

```
%编写目标函数
function [sol, eval] =gaDemo1Eeval(sol,options)
x=sol(1);
eval = x + 10*sin(5*x)+7*cos(4*x);

%参数说明
%eval：个体的适应度；
```

```
%sol：当前个体，n+1 个元素的行向量。

%遗传算法求最大值
clc
fplot('x + 10*sin(5*x)+7*cos(4*x)',[0 9])
% 生成初始种群，大小为 10
initPop=initializega(10,[0 9],'gademo1eeval1');
plot (initPop(:,1),initPop(:,2),'b*')
% 调用遗传函数
% 1 次遗传迭代
[x endPop] = ga([0 9],'gademo1eeval1',[],initPop,[1e-5...
1 1],'maxGenTerm',1,'normGeomSelect',[0.08],['arithXover'], ... [20], 'nonUnifMutation',[2 1 3]);

plot (endPop(:,1),endPop(:,2),'bo')
% 25 次遗传迭代
[x endPop bpop trace] = ga([0 9],'gademo1eeval1',[],initPop, [1e-6 1 1],'maxGenTerm',25,
                    'normGeomSelect',[0.08], ['arithXover'],[2],'nonUnifMutation',[2 25 3]);

plot (endPop(:,1),endPop(:,2),'y*')
figure(2)
plot(trace(:,1),trace(:,3),'y-')
hold on
plot(trace(:,1),trace(:,2),'r-')
xlabel('Generation'); ylabel('Fittness');
legend（'解的变化','种群平均值的变化'）;
```

（1）运行以上程序，目标函数的图形和初始随机种群个体分布图，如图 5-27 所示。

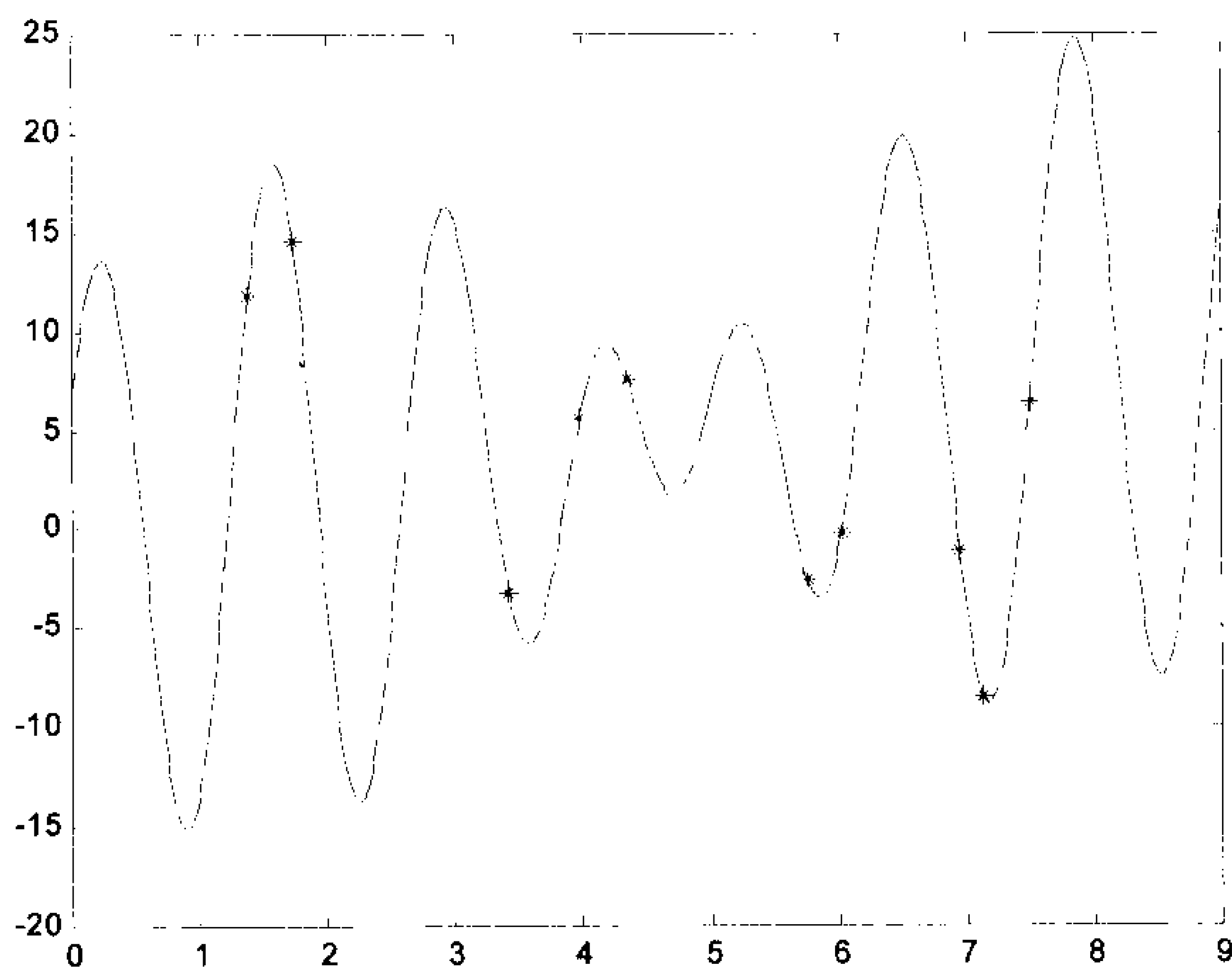


图 5-27 初始种群分布图

（2）经过一次遗传迭代后，寻优结果如图 5-28 所示。

图中“○”代表经过一次迭代后的个体分布，此时最优解为：

```
x =
    8.0217    20.1903
```

（3）经过 25 次迭代后，寻优结果如图 5-29 所示。

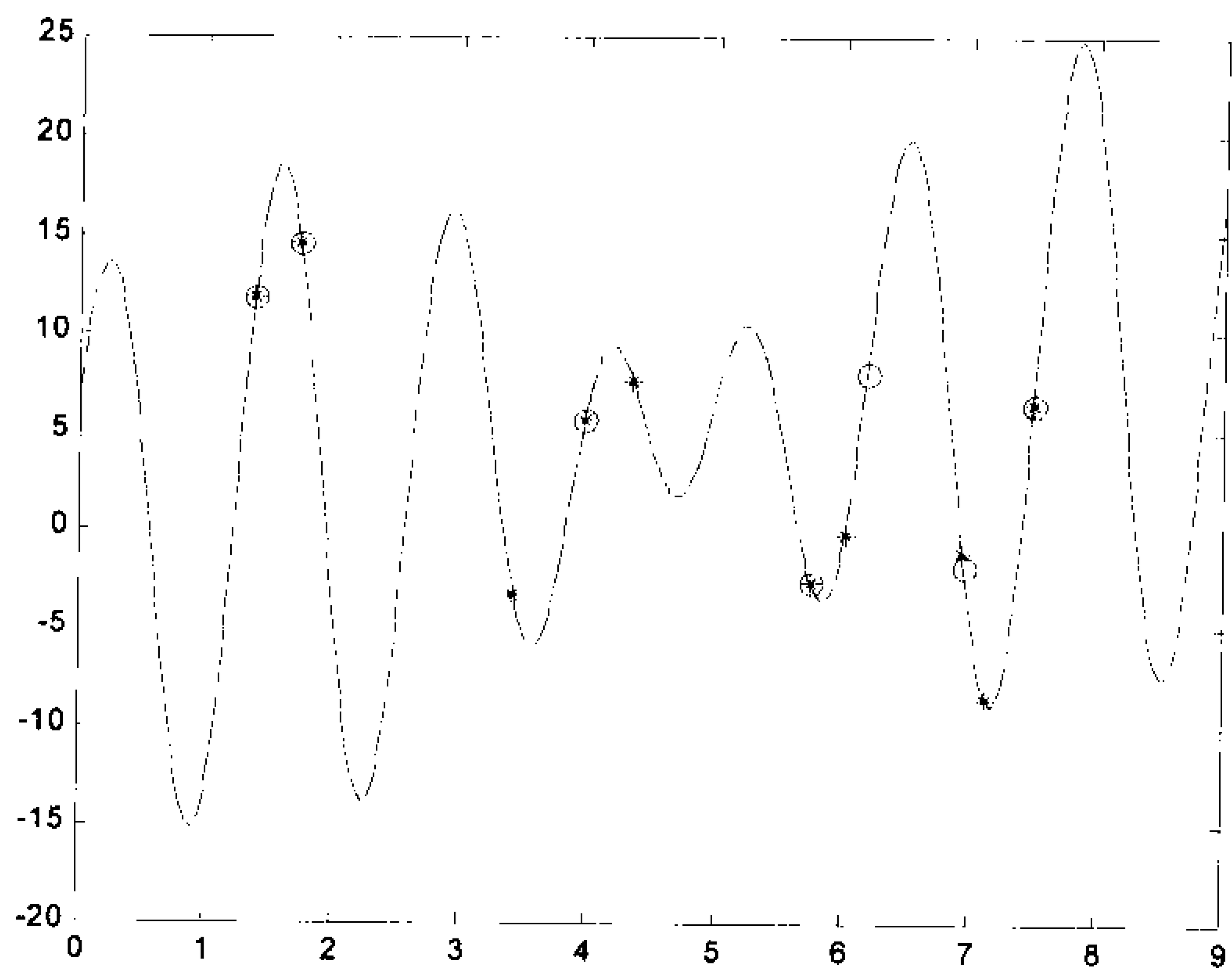


图 5-28　一次迭代寻优结果

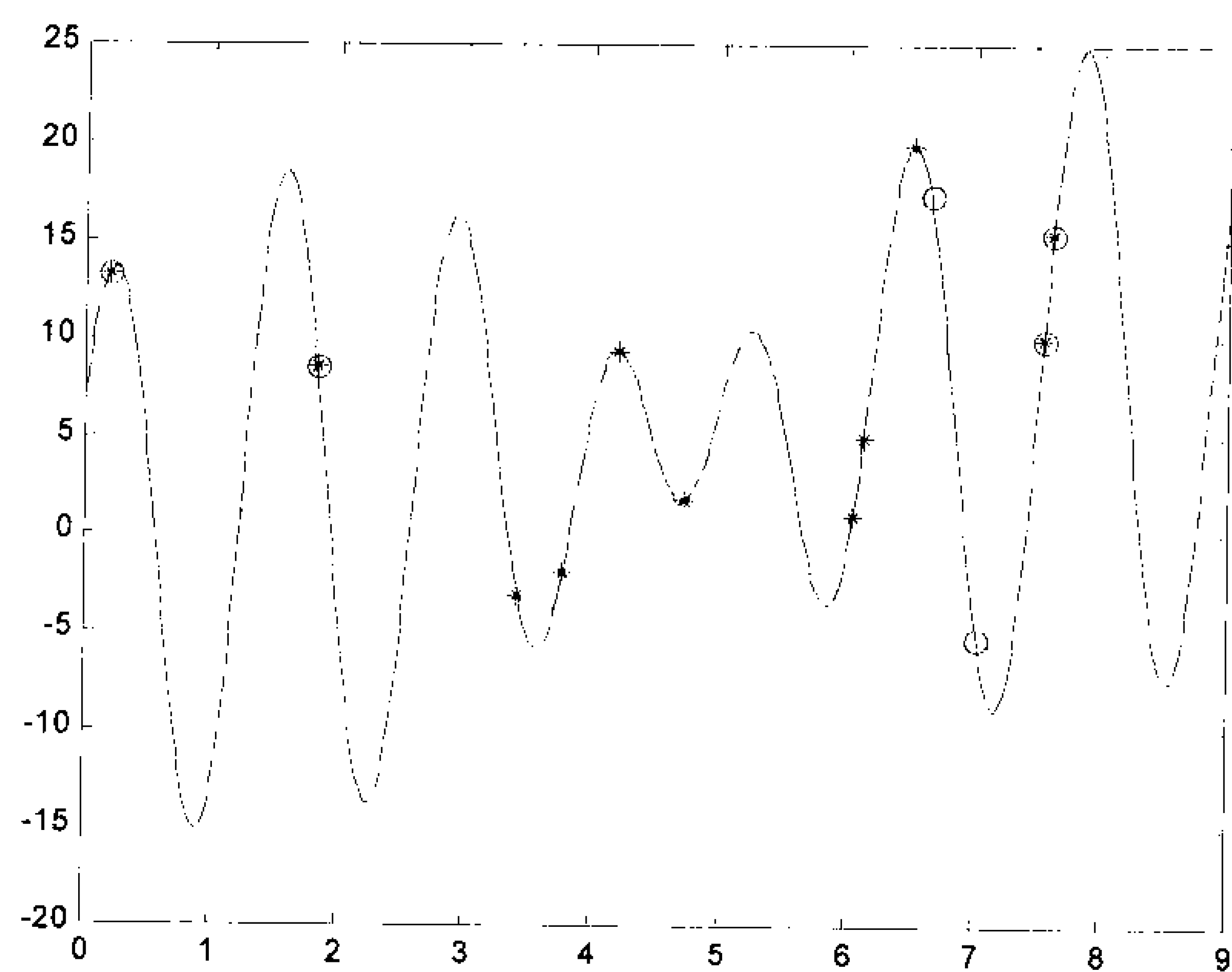


图 5-29　25 次迭代寻优结果

图中“○”代表经过 25 次迭代后的个体分布，此时最优解为：

```
x =
7.8569    24.8554
```

迭代过程如表 5-2 所示。

表 5-2　迭代过程

迭代次数	1	3	6	7	9
函数值	20.190259	20.551384	24.270734	24.421193	24.744628
迭代次数	11	15	16	25	
函数值	24.845763	24.855317	24.855361	24.855361	

即当 x=7.8569 时，$f(x)$取最大值 24.8554。

（4）如图 5-30 所示绘出了遗传算法的寻优性能。

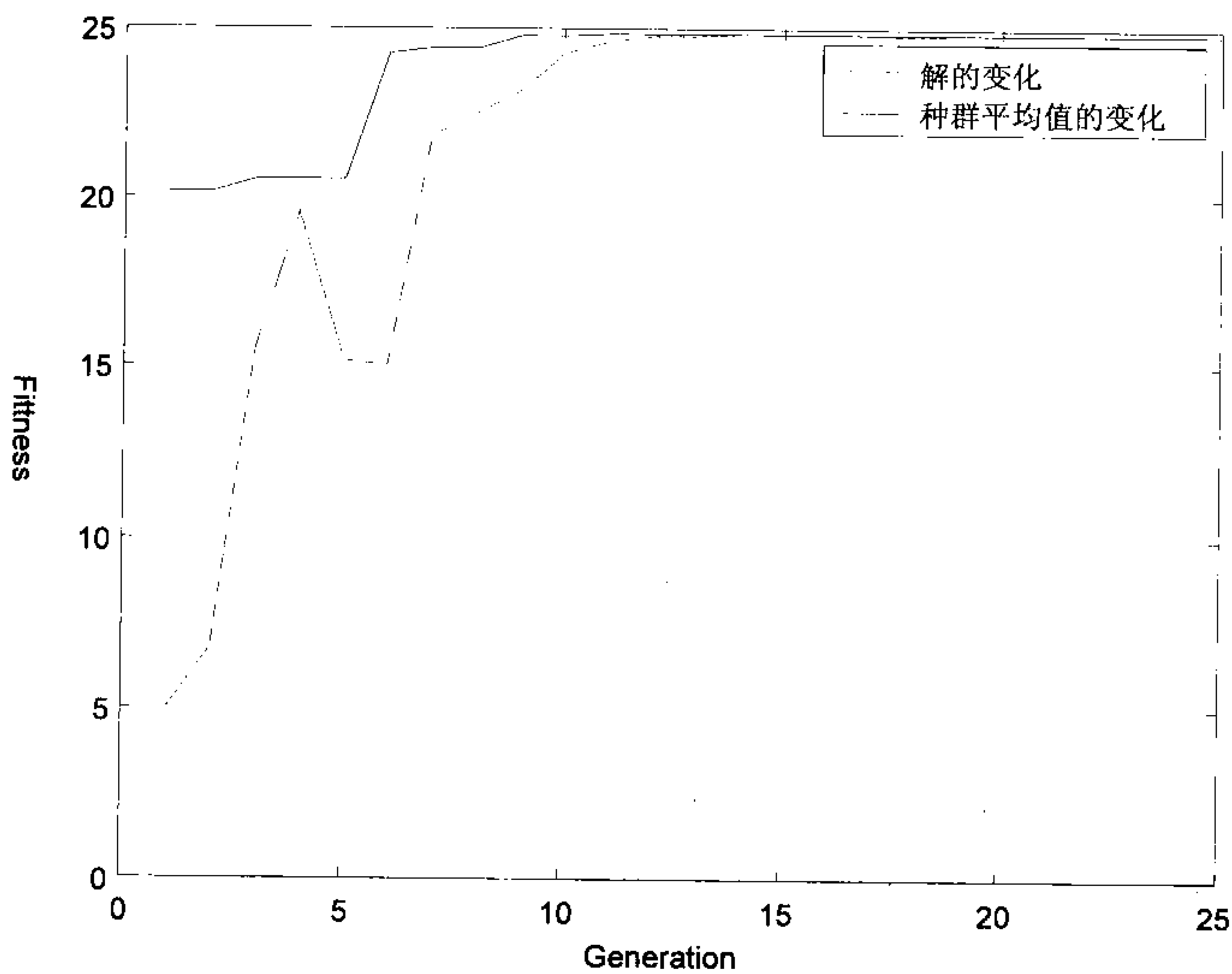


图 5-30　遗传算法的寻优性能

第 6 章　Simulink 仿真高级技术

Simulink 是一种集成交互式图形设计环境，也是 MATLAB 的重要组件之一，它可以实现各种动态系统进行建模、分析和仿真。在 Simulink 环境中，不需要编写程序代码，只需要通过简单直观的鼠标操作，就可以轻而易举地构造出复杂动态系统的仿真模型。正因为如此，Simulink 技术备受青睐，在航空航天动力系统、卫星控制制导系统、通信系统、船舶及汽车等领域已经获得广泛应用。

与 MATLAB R2006b 的 Simulink 6.5 相比，MATLAB R2007a 的 Simulink 6.6 作了很大的改动，修正了此前版本的一些 Bug，新增了 31 项内容。其中最显著的特点是支持信号源、数据录入和显示、数据总线、Real-Time Workshop 代码产生、S-函数（2 级 M 文件的 S-函数）、Stateflow 数据表和 Simulink 模块等事件的多维信号处理。

为了使读者在较短的时间内掌握 Simulink 的基础知识，并精通其高级仿真技术，本章首先介绍了 Simulink 的基本功能、编辑窗口、模块库、模块基本操作和仿真参数的设置等一些入门知识；接着讲述了 Simulink 模型的图形调试器、调试命令行及它们的使用方法；然后阐述了 Simulink 子系统的建立与封装技术，Simulink 模型构造、编辑、仿真命令和回调方法，S-函数的基本概念、工作原理及其用 M 文件、C MEX 文件和创建器编写的方法，同时还叙述了 Stateflow 基本原理、应用基础、常用命令和建模方法；本章最后论述了 Real-Time Workshop 的基本功能，Simulink 模型的普通实时程序和实时代码的生成技术。在讲解以上 Simulink 的入门知识和高级仿真技术的过程中，辅佐了丰富的实例。读者可以通过这些实例，迅速掌握 Simulink 仿真技术的精髓。

本章主要内容：

- Simulink 入门知识
- Simulink 模型的调试方法
- Simulink 子系统的建立及封装技术
- Simulink 仿真命令及回调方法
- S 函数的 M 文件、C MEX 文件和创建器的编写方法
- Stateflow 应用基础和建模方法
- Real-Time Workshop 普通实时程序和实时代码的生成技术

6.1 Simulink 入门

6.1.1 Simulink 功能

Simulink 是一种图形化仿真工具包，能够进行动态系统建模、仿真和综合分析，可以处理线性和非线性系统，离散、连续和混合系统，以及单任务和多任务系统，并在同一系统中支持不同的变化速率。

1．交互式仿真工具

Simulink 具有非常高的开放性，提倡将模型通过框图形式表示出来，或者将已有的模型添加组合到一起，又或者将自己创建的模块添加到模型当中。Simulink 具有较高的交互性，允许随意修改模块参数，并且可以直接无缝地使用 MATLAB 的所有分析工具。分析得到的结果，并进行可视化显示。Simulink 的一个意图就是让用户在使用 Simulink 的同时能够感受到建模与仿真的乐趣。Simulink 提供了大量的模块，方便用户快速地建立动态系统模型，只需要操作鼠标，就能够建立非常复杂的仿真模型，对模型中的连接数量和规模没有限制。Simulink 应用领域也非常广泛，包括航空航天、电子、力学、数学、通信、影视和控制等。世界各地的工程师都在利用它建立实际工程问题的模型。

2．图形化动力系统建模工具

利用 Simulink 工具包可以不受线性系统模型的限制，能够建立更加真实的非线性系统，如在系统中考虑摩擦力、空气阻力、齿轮滑动等。它将计算机变成一个系统建模与分析的验室，特别是对于那些无法做实验的系统（如客机机翼的颤动、生物链和货币供给等系统）更是如此。

Simulink 提供了非常全面的模块库及工具箱，使得模型的建立十分方便。对于模型的建立，可以采用从上到下或者从下到上的顺序，也可以按照信号流程的方式。后面一种建模方法的思路清晰，对模块的相互作用和组织形式一目了然。

在建立好模型之后和运行仿真之前，必须对模型进行参数设置。仿真所需要的模型参数设置可以通过 MATLAB 命令或者 Simulink 菜单进行。这两种方法各有千秋，前者适合批处理多个仿真，而后者则直观方便。模块参数的设置方法是双击相应模块，在弹出对话框中进行参数设置。仿真完成后，可以使用 scope 或 XY Graph 等模块来显示结果，还可以直接将结果输出到 MATLAB 的工作空间，然后利用各种图像处理函数来处理结果。Simulink 除了能够将数据导出到 MATLAB 工作空间之外，还可以将 MATLAB 工作空间中的数据导入到 Simulink 模型中。

3．Simulink 的扩展功能

Simulink 是一个开放式结构体系，允许用户自己开发各种功能的模块，无限制地添加到 Simulink 环境中，以满足不同任务的要求。

可以通过以下方式来增强 Simulink 的模块功能。

- 采用 MATLAB 的 M 文件、Fortran 以及 C 代码生成自定义模块。
- 利用 Simulink 本身来建立子系统，封装自定义的模块。
- 将 Simulink 与开发好的 S-函数无间隙连接起来，完成复杂的功能。
- 将原有的 Fortran 和 C 代码连接起来。
- 第三方开发的工具箱。
- 其他工程软件(如 Adams、Femlab 和 Labview 等)与 MATLAB 的衔接接口，Simulink 可以非常方便地使用这些软件中的信息，同时也可以被这些软件调用。

4．Simulink 专用模块库与相关产品

为丰富 Simulink 建模系统，MathWorks 公司开发或收购了许多有特殊功能的模块程序包。利用这些功能强大的程序包，使得用户能够非常方便地建立模型或者完成系统分析，生成建立的 Simulink 模型实时代码。

常用的 Simulink 模块（Blockset）如下。

- CDMA Reference Blockset（CDMA 通信系统设计与分析）
- Communication Blockset（通信系统工具箱）
- Dials & Gauges Blockset（交互式图形和控制面板设计工具箱）
- DSP Blockset（数字信号处理工具箱）
- Fixed-Point Blockset（定点运算控制系统工具箱）
- Motorla DSP Developer'S Kit（Motorola DSP 开发工具）
- Neural Network Blockset（神经网络）
- Nonlinear Control Design Blockset（非线性控制设计工具箱）
- Real Time Workshop（实时系统）
- SimMechanics（机构仿真）
- SimPowerSystems（电力电动工具箱）
- Stateflow（流程控制）
- TI DSP Developer's Kit（TI DSP 开发工具）

6.1.2　Simulink 运行方法及编辑窗口

1．Simulink 运行方法

Simulink 的运行方法有以下 3 种。

- 在 MATLAB 的命令窗口中输入“Simulink”指令。
- 单击 MATLAB 工具栏上的“Simulink”图标。
- 在 MATLAB 菜单中执行【File】→【New】→【Model】命令。

执行第一种方法或第二种方法后，将弹出如图 6-1 所示的 Simulink 模块库浏览器（在该浏览器中可以找到大部分常用的模块）；执行第三种方法后，将弹出如图 6-2 所示的新建模型窗口。单击图 6-1 中工具栏左边的图标，也会弹出如图 6-2 所示的窗口。

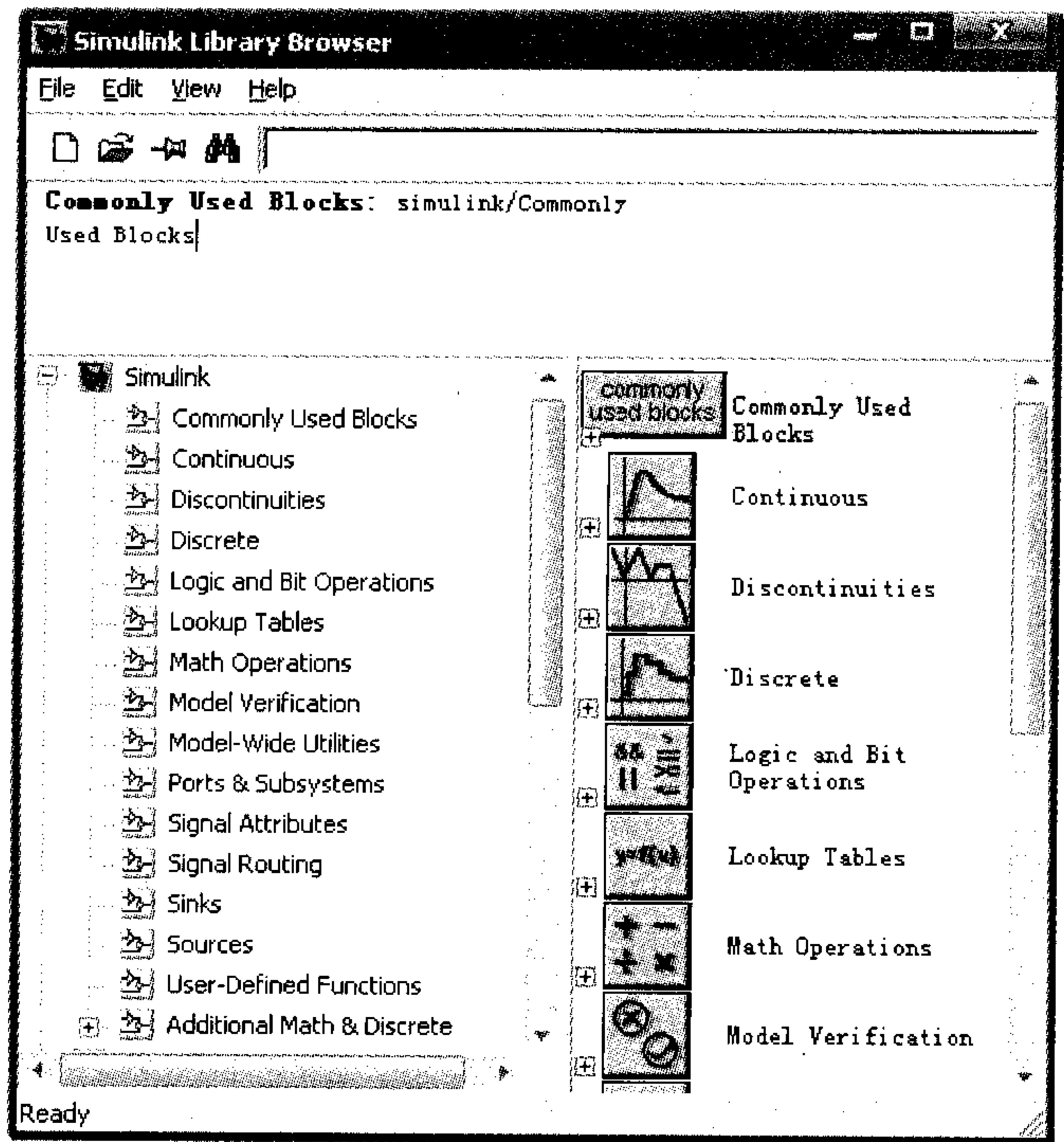


图 6-1 Simulink 模块库浏览器

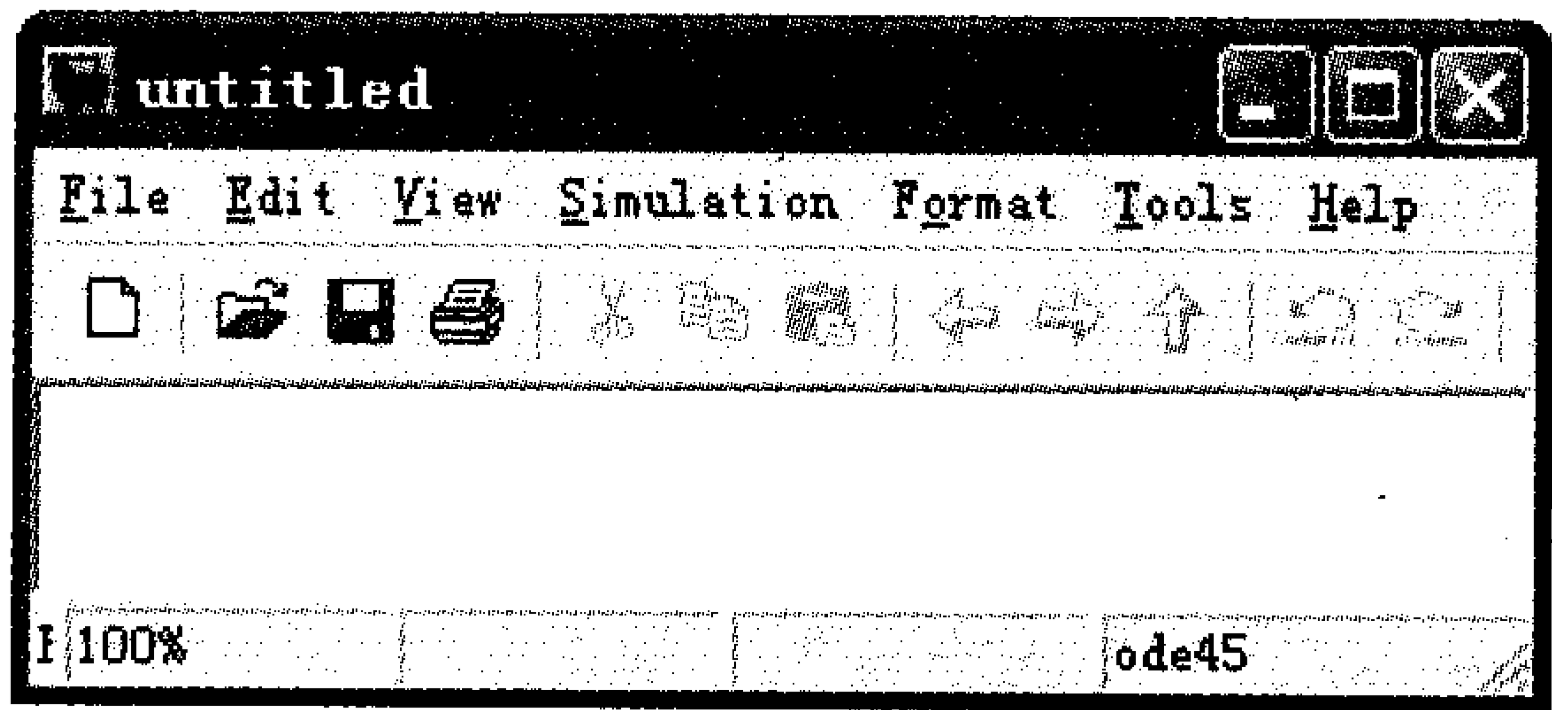


图 6-2 新建模型窗口

打开已经存在的模型文件也有以下 3 种方法。

- 直接在 MATLAB 命令窗口输入模型文件名（不要加扩展名“.mdl”），这要求文件在当前的路径范围内。
- 在 MATLAB 菜单上执行【File】→【Open】命令，在弹出的浏览窗口中选择所需的模型文件名。
- 单击图 6-1 中的图标。

2．Simulink 编辑窗口

图 6-2 所示的新模型窗口有两种表现形式，图 6-2 是“单窗口”表现形式，单击图标，可切换成如图 6-3 所示的“双窗口”形式。它的左窗口为“模型浏览器（Model Browser）”，用来展示该模型的“分层”子系统目录；而右窗口仍展示相应系统的连接方块图。整个新建模型窗口的组成是：菜单栏、工具栏、编辑窗口和状态栏。

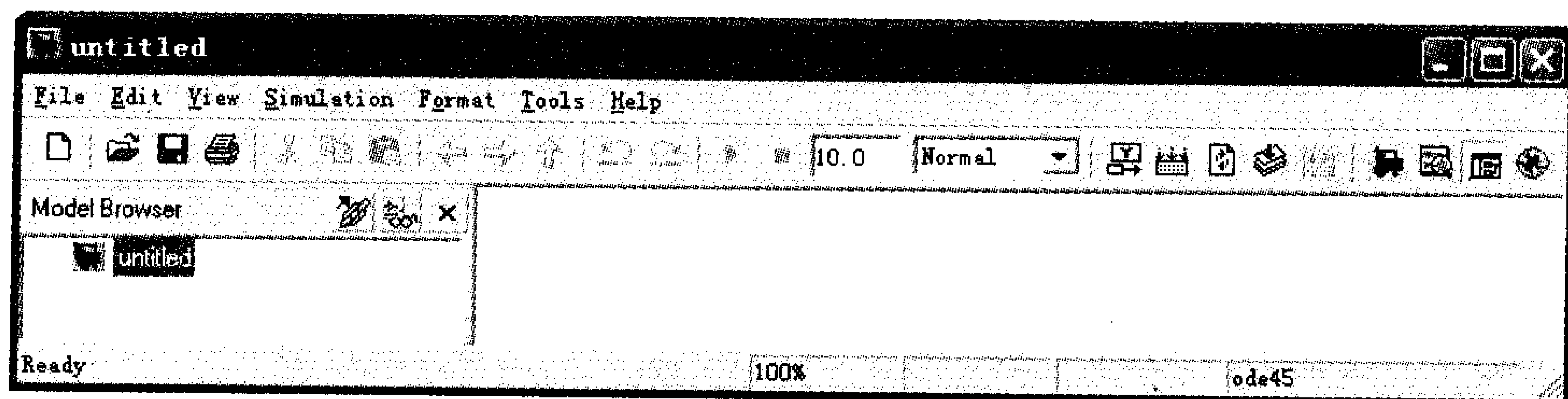


图 6-3 展现浏览器的模型窗口

1）工具栏

最左边的 9 个图标实现标准的 Windows 操作，其余图标可用于库浏览器的打开、调试过程控制、浏览器单双窗口外形切换、显示库链接和观察封装子系统等。

2）状态栏

以图 6-3 为例，自左向右的文字分别表示："Ready" 表示模型已准备就绪而等待仿真指令；"100%" 表示编辑框模型的显示比例；"ode45" 表示仿真所选用的积分解算器。此外，在仿真过程中，状态栏的空白格处还会出现动态信息。

3）菜单栏

Simulink 专用菜单项如表 6-1 所示。

表 6-1 Simulink 专用菜单项

【File】:

名　称	功　能	名　称	功　能
New：Model	新建模型	Preferences	Simulink 界面形态的默认设置选项
Model properties	模型属性	Print Details	打印模型细节设置

【Edit】:

名　称	功　能	名　称	功　能
Paste Duplicate Import	粘贴复制的输入模块	Look Under Mask	打开精装子系统的内部结构
Copy Model To Clipboard	将模型复制到剪贴板	Refresh Model Blocks	刷新输入、输出和参数设置
Create Subsystem	创建子系统	Update Diagram	更新模型框图的外观
Mask Subsystem	精装子系统		

【View】:

名　称	功　能	名　称	功　能
Go To Parent	展示其直接父系统	Library Browser	激活库浏览器
Toolbar	显示工具栏	Zoom In	使显示模型放大
Statebar	显示状态栏	Zoom Out	使显示模型变小
Model Brower Options	展示模型浏览器	Fit System To View	自动选择最合适的显示比例
Block Data Tips Options	鼠标位于模块上方时，显示模块内部数据	Normal	以正常比例显示模型
System Requirements	系统要求设置		

【Simlulation】:

名　称	功　能	名　称	功　能
Start/Stop	启动/停止仿真	Accelerator	产生模型的实时运行程序
Configuration Parameters	设置仿真参数	External	建立外部仿真模型
Normal	普通 Simulink 模型		

【Format】:

名　称	功　能	名　称	功　能
Font	字体设置	Show Port Labels	显示端口标签
Text Alignment	标注文字对齐工具	Foreground Color	前景颜色，从子菜单中选择
Enable Tex Commands	将 Tex 控制命令设置生效	Background Color	后景颜色，从子菜单中选择
Filp Name	向对侧搬动模块名	Screen Color	屏幕颜色
Filp Block	翻转模块	Port/Signal Displays	端口/信号显示
Rotate Block	顺时针旋转模块	Block Displays	模块显示
Show Name	显示模块名	Library Link Display	库链接显示
Show Drop Shadow	显示阴影效果		

【Tools】菜单包含了 MATLAB Simulink 提供的附属工具包。这个菜单的命令将取决于用户安装 MATLAB 时的选择。

6.1.3　Simulink 模块库

熟悉 Simulink 模块库所包含的内容是建立模型的基础，只有熟练地掌握了模块库才能够让用户快速地建立模型，以最少的模块来建立模型，或者以最快仿真速度为目的来建立模型。

从图 6-1 可看出，Simulink 模型库浏览器窗口中包含了丰富的模块组，主要的模块组如下。

- 常用模块组（Commonly Used Blocks）
- 连续模块组（Continuous）
- 非连续模块组（Discontinuities）
- 离散模块组（Discrete）
- 逻辑和位操作模块组（Logic and Bit Operations）
- 寻表操作组（Lookup Tables）
- 数学操作模块组（Math Operations）
- 模型确认操作模块组（Model Verification）
- Model-Wide Utilities
- 端口与子系统模块组（Ports ＆ Subsystems）
- 信号属性（Signal Attributes）
- 信号路由模块组（Signal Routing）

- 接收器模块组（Sinks）
- 信号源模块组（Sources）
- 自定义函数模块组（User-Defined Functions）
- 附加操作组（Additional Math & Discrete）

此外，用户还可以自定义模块组。

在浏览 Simulink 模块组时，可以采用如下的方法显示模块组的全貌：用鼠标右键单击相应的模块组名，执行弹出菜单的命令（如【Open the Continuous library】），这样就可以弹出相应的模块组界面。

为了节省篇幅，这里只对 Simulink 中比较重要和常用的模块组及其功能进行说明。

1．常用模块组

常用模块组如图 6-4 所示，它的模块均由其他模块组中的模块构成，主要是为了方便用户调用最常用的模块，以提高建模的速度。该模块组中模块的功能将在其他模块组介绍。

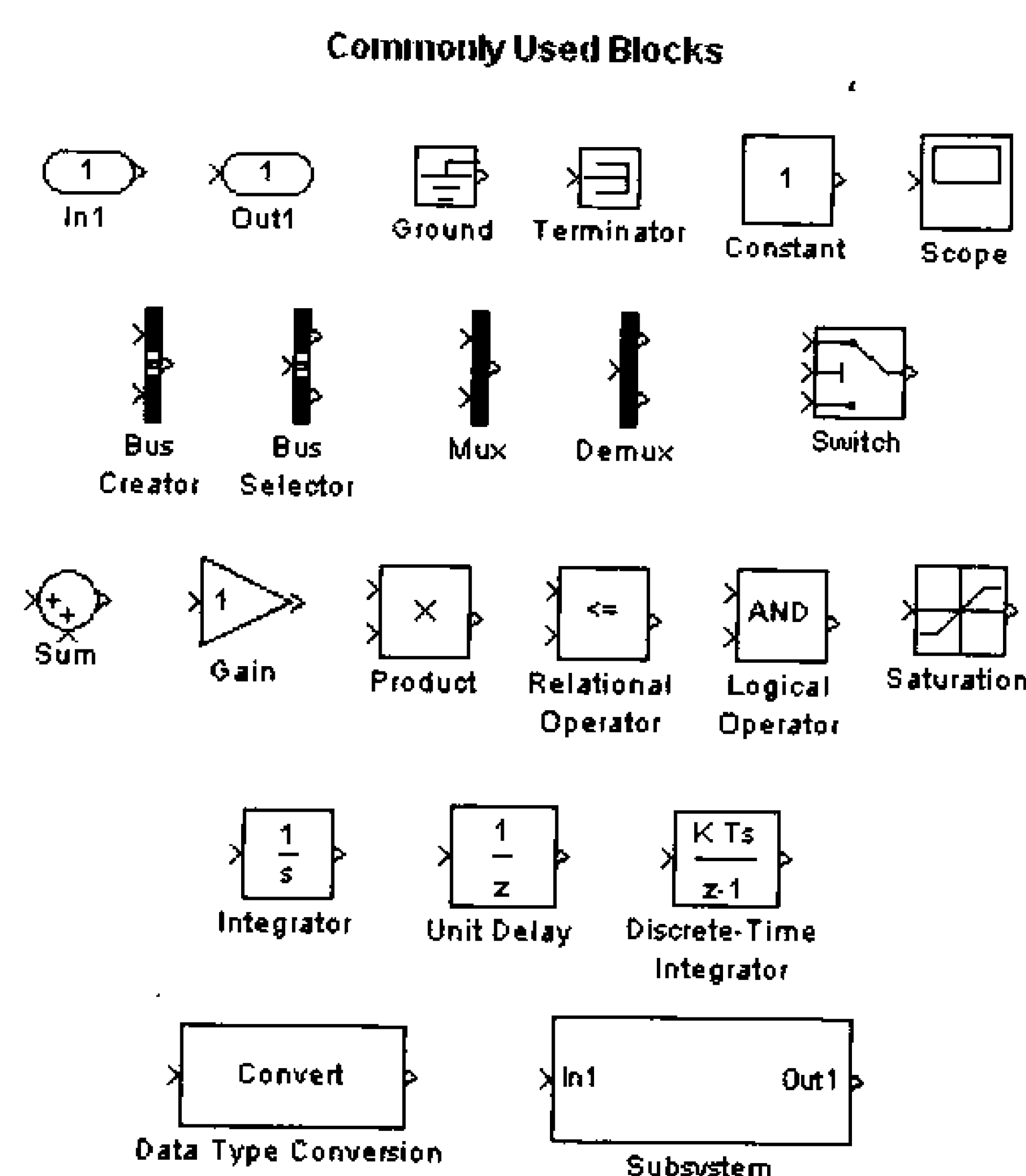


图 6-4　常用模块组

2．连续模块组

连续模块组如图 6-5 所示，它包含常用的连续系统模型中所涉及的模块，这些模块及其功能如表 6-2 所示。

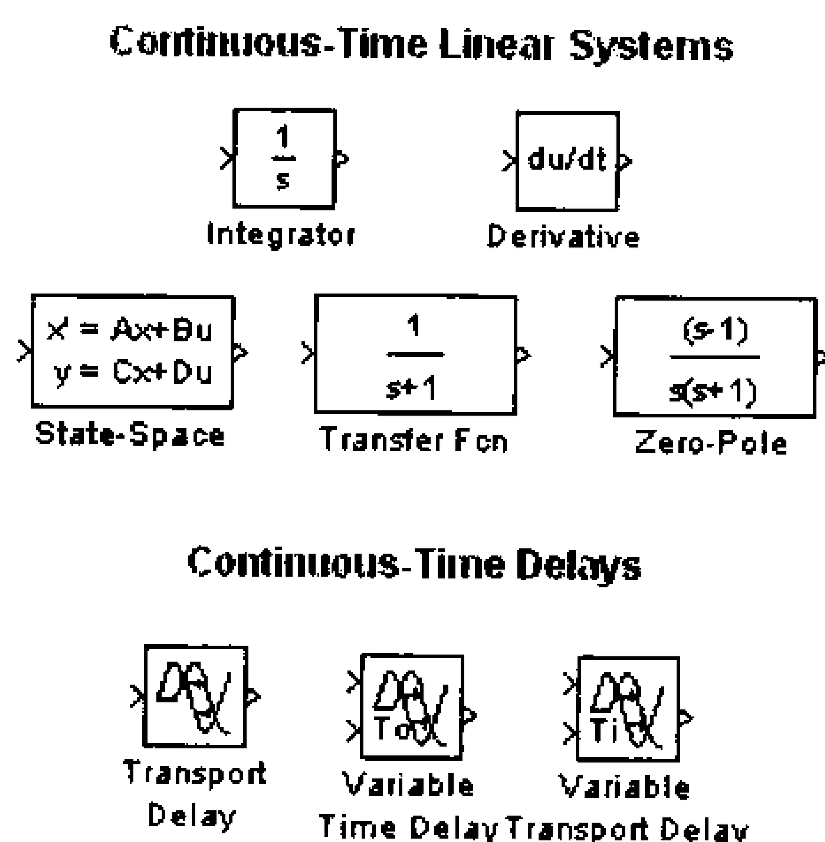


图 6-5　连续模块组

表 6-2　连续模块及其功能

名　　称	功　　能	名　　称	功　　能
Integrator	积分	Derivative	微分
State-Space	状态空间	Transfer Fcn	传递函数
Pole-Zero	零极点	Transport Delay	时间延迟
Variable Transport Delay	可变时间延迟		

3．离散模块组

离散系统在系统仿真中被广泛使用，基于这种考虑，MathWorks 公司开发并推出了离散系统模块库，为离散系统仿真提供便利。离散模块组如图 6-6 所示，其常用的模块及功能如表 6-3 所示。

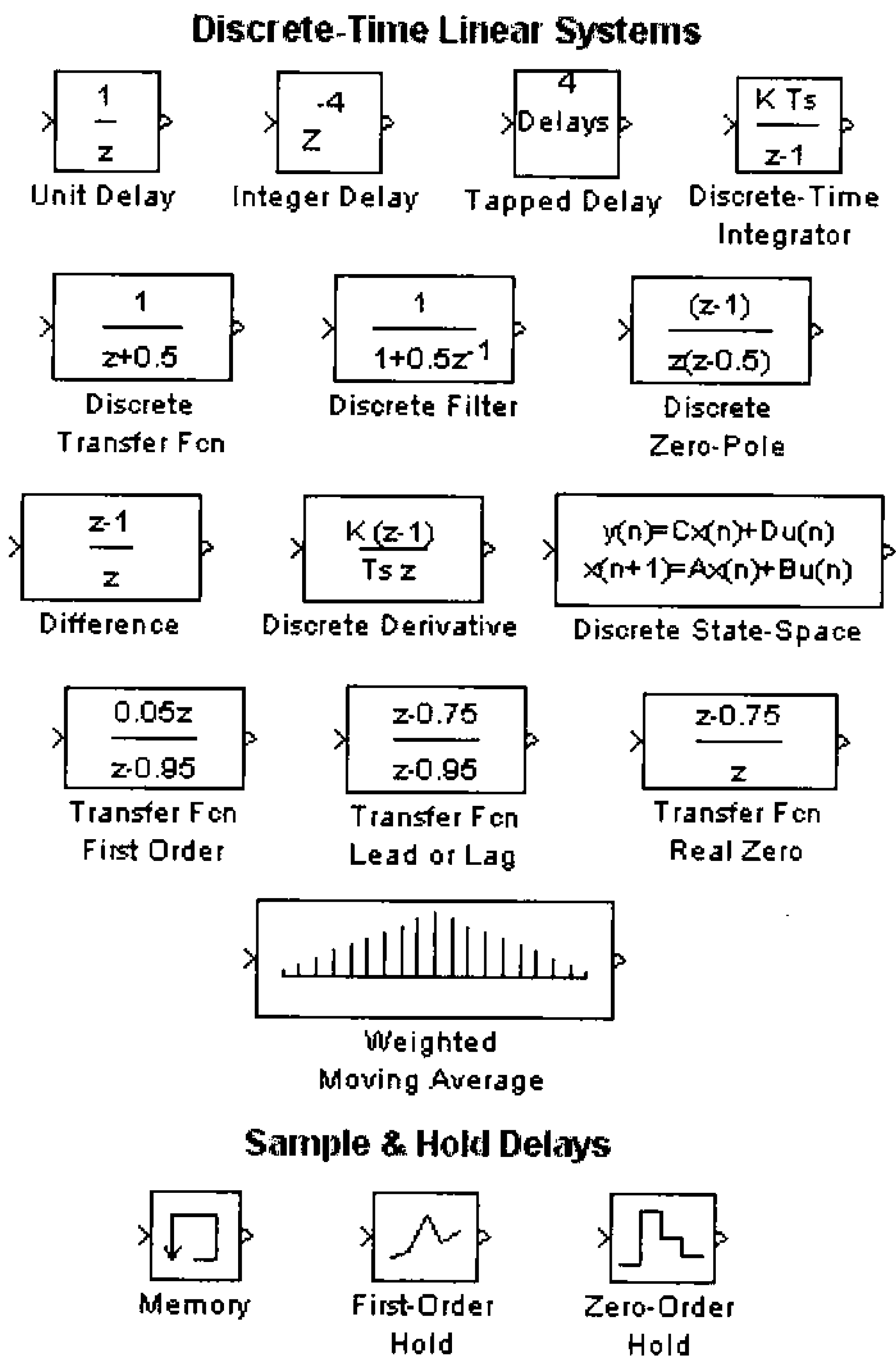


图 6-6　离散模块组

表 6-3　离散模块及其功能

名　　称	功　　能	名　　称	功　　能
Unit Delay	采样保持，延迟一个周期	Discrete Derivative	离散派生方程
Integer Delay	采样保持，延迟一个整数周期	Discrete－Time Integrator	离散时间积分
Discrete Transfer Fcn	离散传递函数	Zero-Order Hold	零阶保持器
Discrete filter	离散滤波器（IIR,FIR）	Discrete Zero-Pole	离散零-极点模型
Discrete State-Space	离散状态方程	First-Order Hold	一阶保持器

4．逻辑和位操作模块组

逻辑和位操作模块组如图 6-7 所示，它包含常用的逻辑运算和关系运算模块，这些模块及其功能分别如表 6-4 和表 6-5 所示。

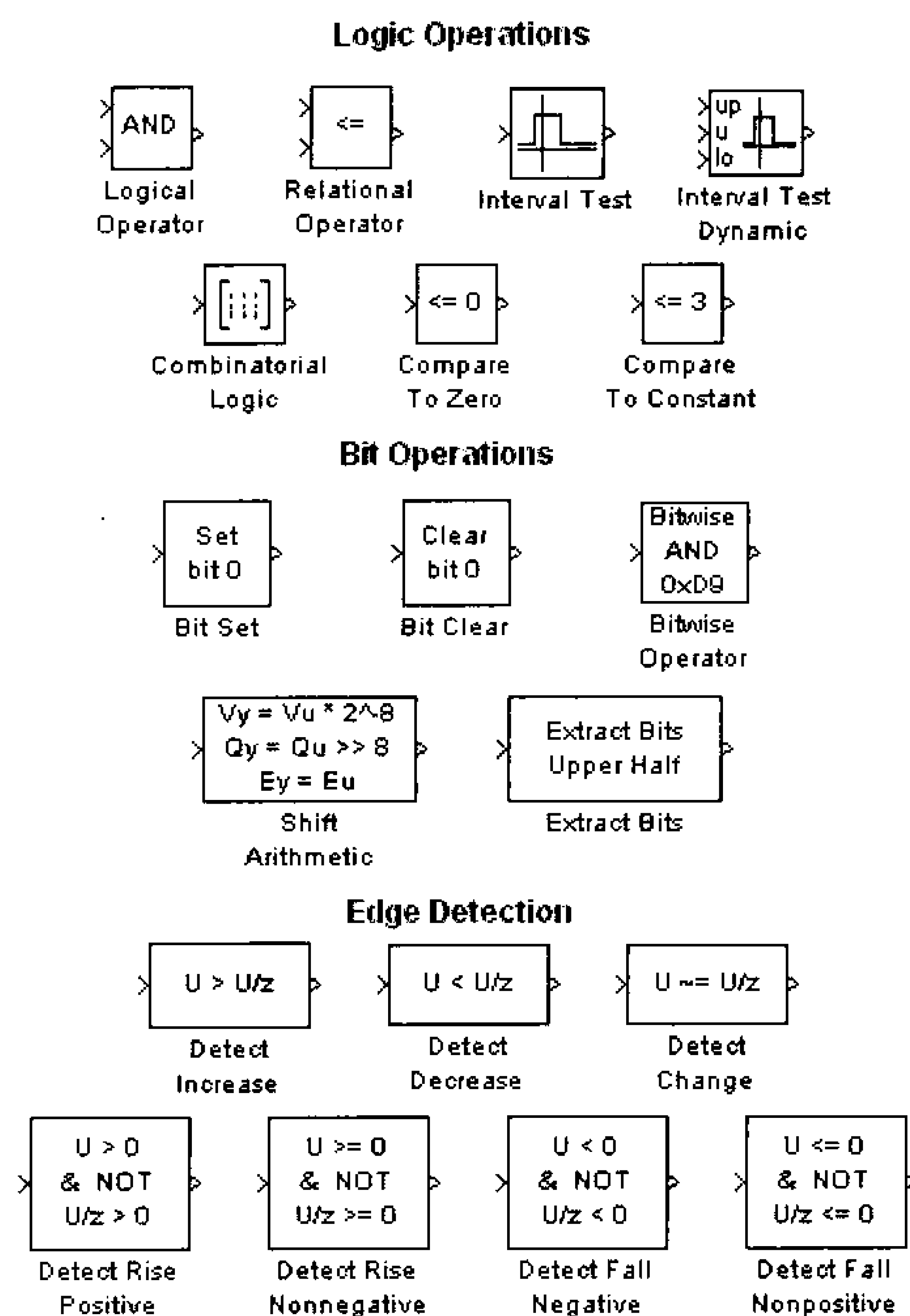


图 6-7 逻辑和位操作模块组

表 6-4 逻辑运算模块及其功能

运 算 模 块	功　　能	运 算 模 块	功　　能
AND	输入全部为真时输出为真	OR	输入有一个为真时输出为真
NAND	输入有一个为非时输出为真	NOR	输入全部为非时输出为真
XOR	输入中有奇数个输入为真时输出为真	NOT	输入为非时输出为真

表 6-5 关系运算模块及其功能

运 算 模 块	功　　能	运 算 模 块	功　　能
==	第一个输入等于第二个输入时为真	~=	第一个输入不等于第二个输入时为真
<	第一个输入小于第二个输入时为真	<=	第一个输入小于或等于第二个输入时为真
>=	第一个输入大于或等于第二个输入时为真	>	第一个输入大于第二个输入时为真

5．寻表模块组

寻表模块组如图 6-8 所示，它包含的常用模块及功能如表 6-6 所示。

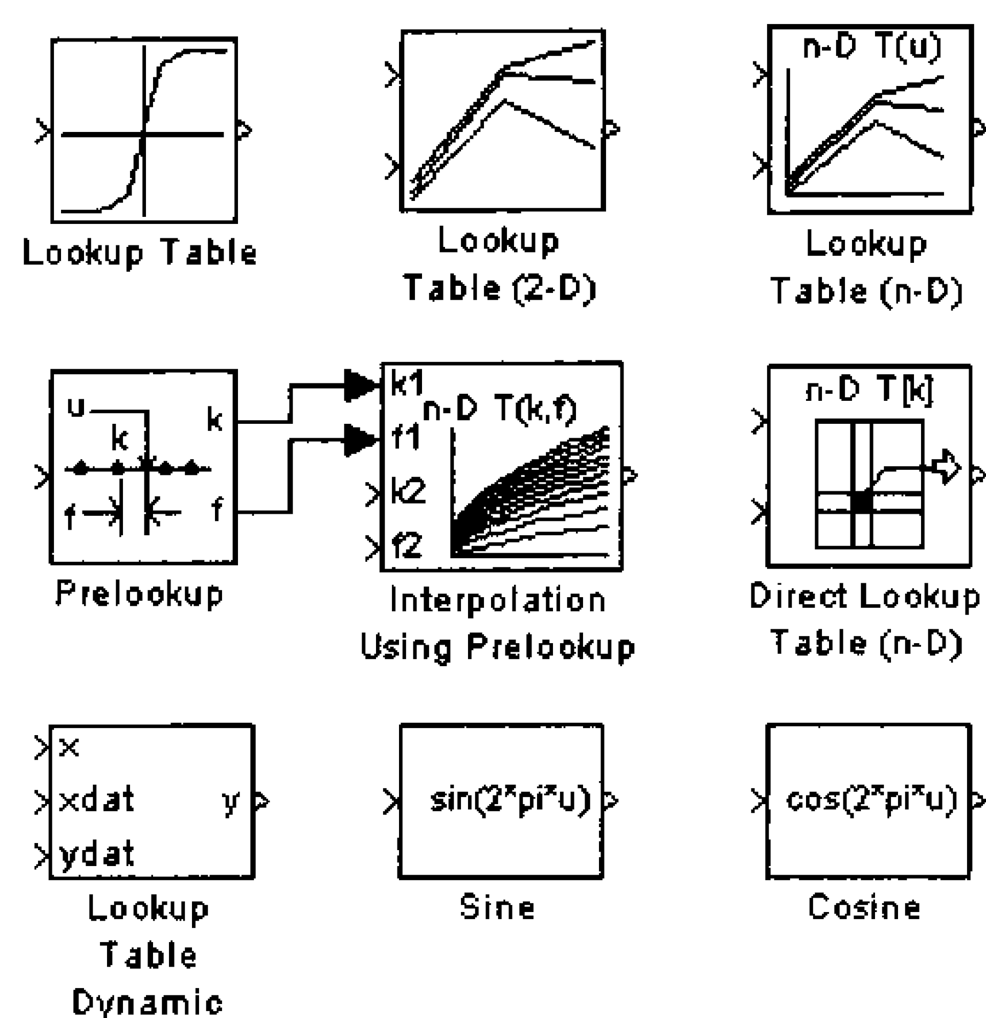


图 6-8　寻表模块组

表 6-6　寻表模块及其功能

名　　称	功　　能	名　　称	功　　能
Look-Up Table	一维查表	Look-up Table(2-D)	二维查表
Lookup Table Dynamic	动态查表	Direct Lookup Table(n~D)	直接查表

6．数学运算模块组

数学运算模块组如图 6-9 所示，它将许多数学运算封装打包，使得数学运算操作变得简单容易，为用户减少了很多程序设计的步骤。数学运算模块组的常用模块及功能如表 6-7 所示。

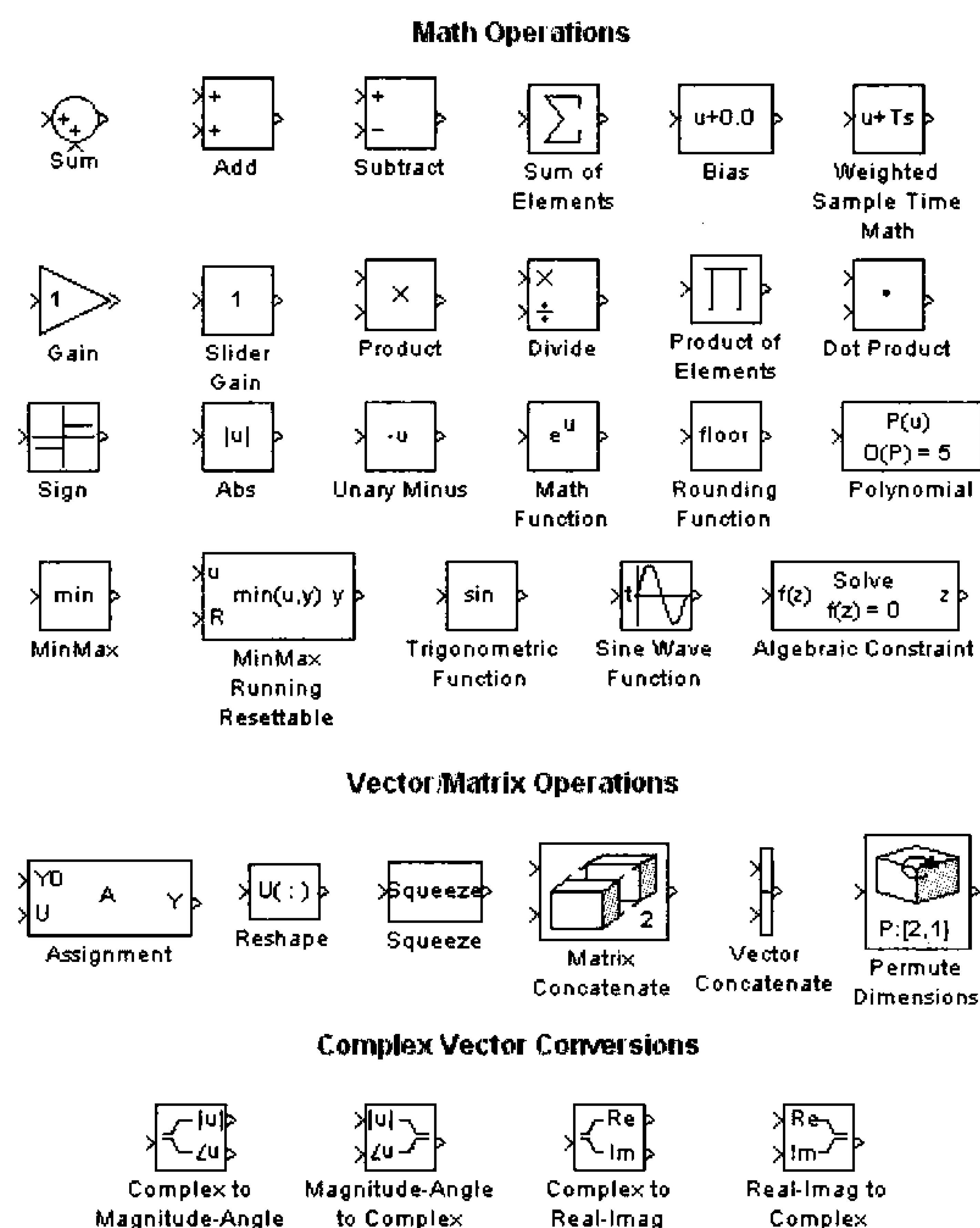


图 6-9　数学运算模块组

表 6-7 数学运算模块及其功能

名　称	功　能	名　称	功　能
Sum	对输入求代数和	Rounding Function	取整函数
Gain	常量增益（输入为一个常数）	MinMax	求最值
Slider Gain	可以用滑动条改变的增益	Trigonometric Function	三角函数
Product	对输入求积或商	Algebraic Constraint	强制输入信号为零
Dot Product	点积（内积）	Complex to Magnititude-Angle	求复数的幅值、相角
Sign	取输入的正负符号	Magnititude-Angle to Complex	根据幅值，相角得到复数
Abs	求输入的绝对值或求模（复数）	Complex to Real-Imag	求复数的实部、虚部
Math Function	数学运算函数	Real-Imag to Complex	根据实部、虚部求复数

7. 端口与子系统模块组

端口与子系统模块组如图 6-10 所示，其模块及功能如表 6-8 所示。

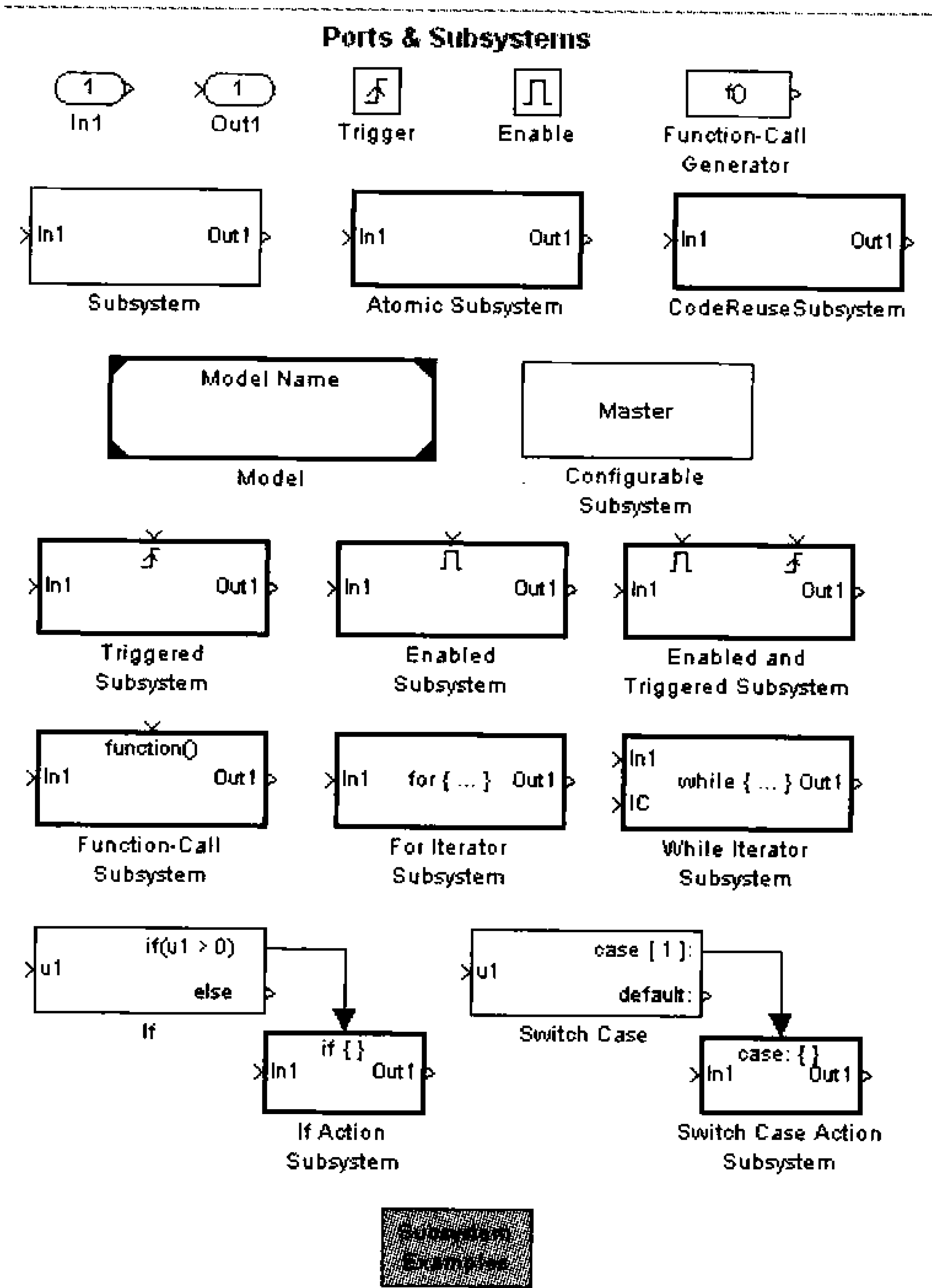


图 6-10 端口与子系统模块组

表 6-8 端口与子系统模块及其功能

名　称	功　能	名　称	功　能
subsystem，Atomic Subsystem，CodeReuse Subsystem	空白子系统	Triggered Subsystem	触发子系统
For Iterator Subsystem，While Iterator Subsystem	结构控制子系统	If Action Subsystem，Switch Case Action Subsystem	转向控制子系统

8. 信号通道模块组

信号通道模块组如图 6-11 所示，其模块及功能如表 6-9 所示。

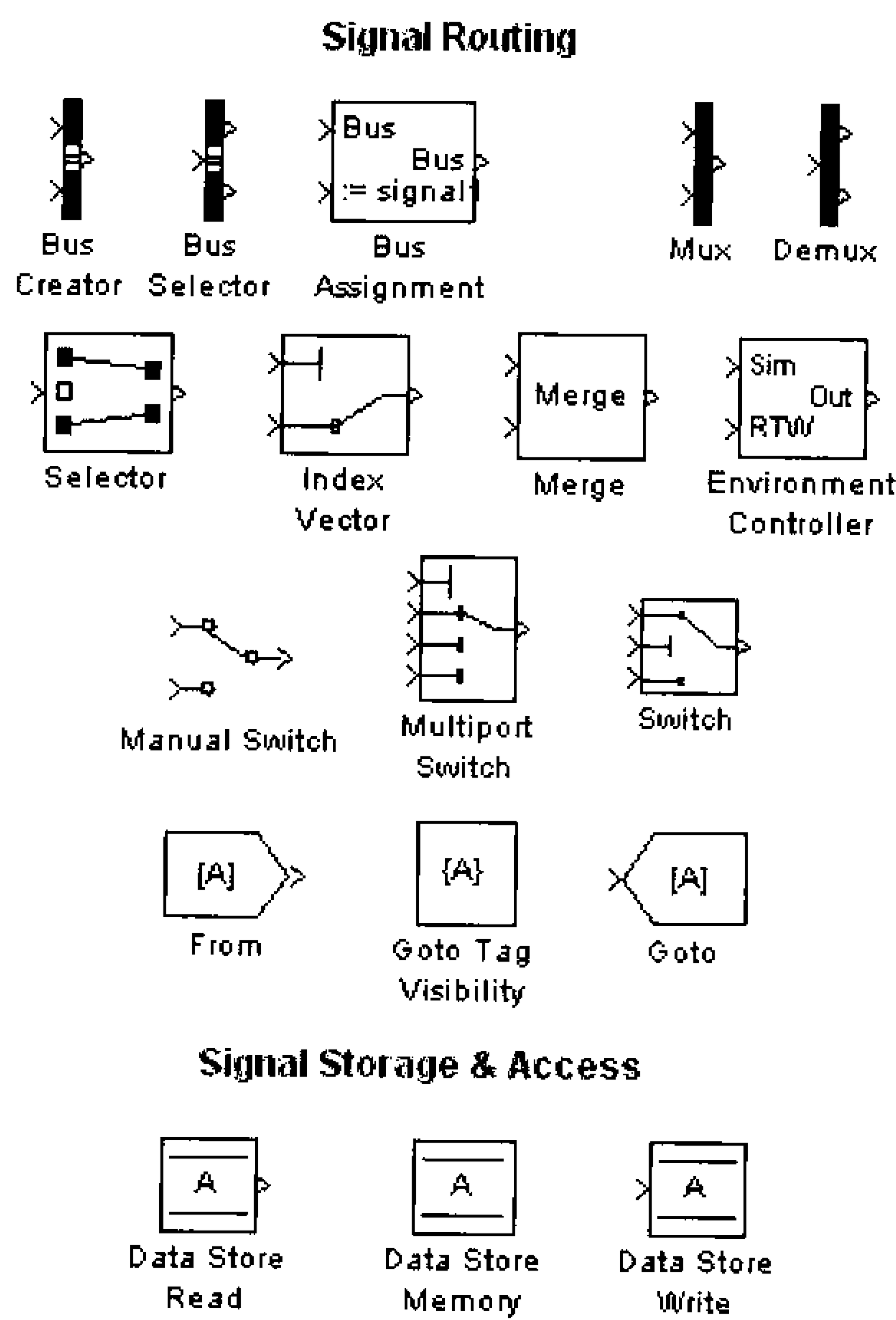


图 6-11　信号通道模块组

表 6-9　信号通道模块及其功能

名　　称	功　　能	名　　称	功　　能
Mux	混路器	Demux	分路器
Manual Switch	手工切换器	Switch	切换器
Multipor Switch	多端口切换器	Data Store Read，Data Store Memory，Data Store Write	数据读取、存储和写入内存

9. 信号接收模块组

信号接收模块组如图 6-12 所示，其模块及功能如表 6-10 所示。

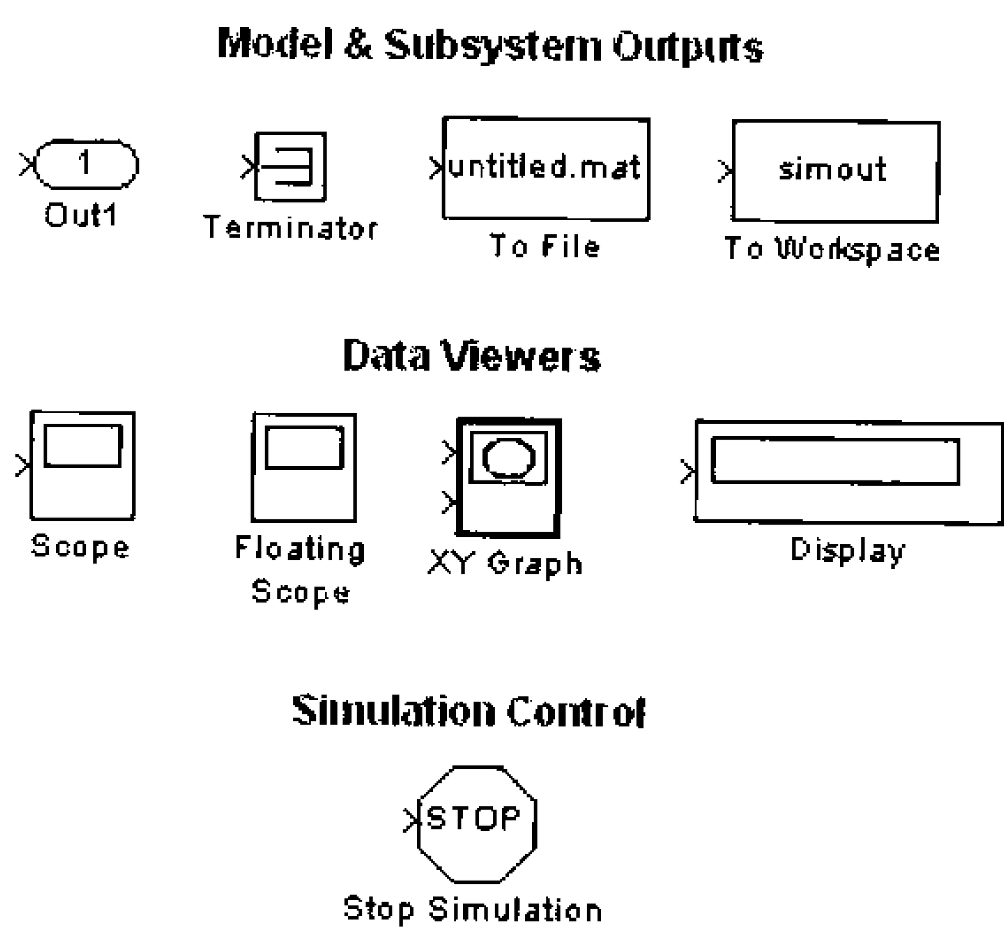


图 6-12　信号接收模块组

表 6-10 信号接收模块及其功能

名 称	功 能	名 称	功 能
Out1	输入端口	Floating Scope	游离示波器
Terminator	接收终端	XY Graph	两个信号的关系图，用 MATLAB 图形显示
To File	保存到文件	Display	实时数值显示
To Workspace	输出到当前工作空间的变量	Stop Simulation	输入不为零时停止仿真
Scope	示波器		

10. 信号源模块组

信号源模块组如图 6-13 所示，其模块及功能如表 6-11 所示。

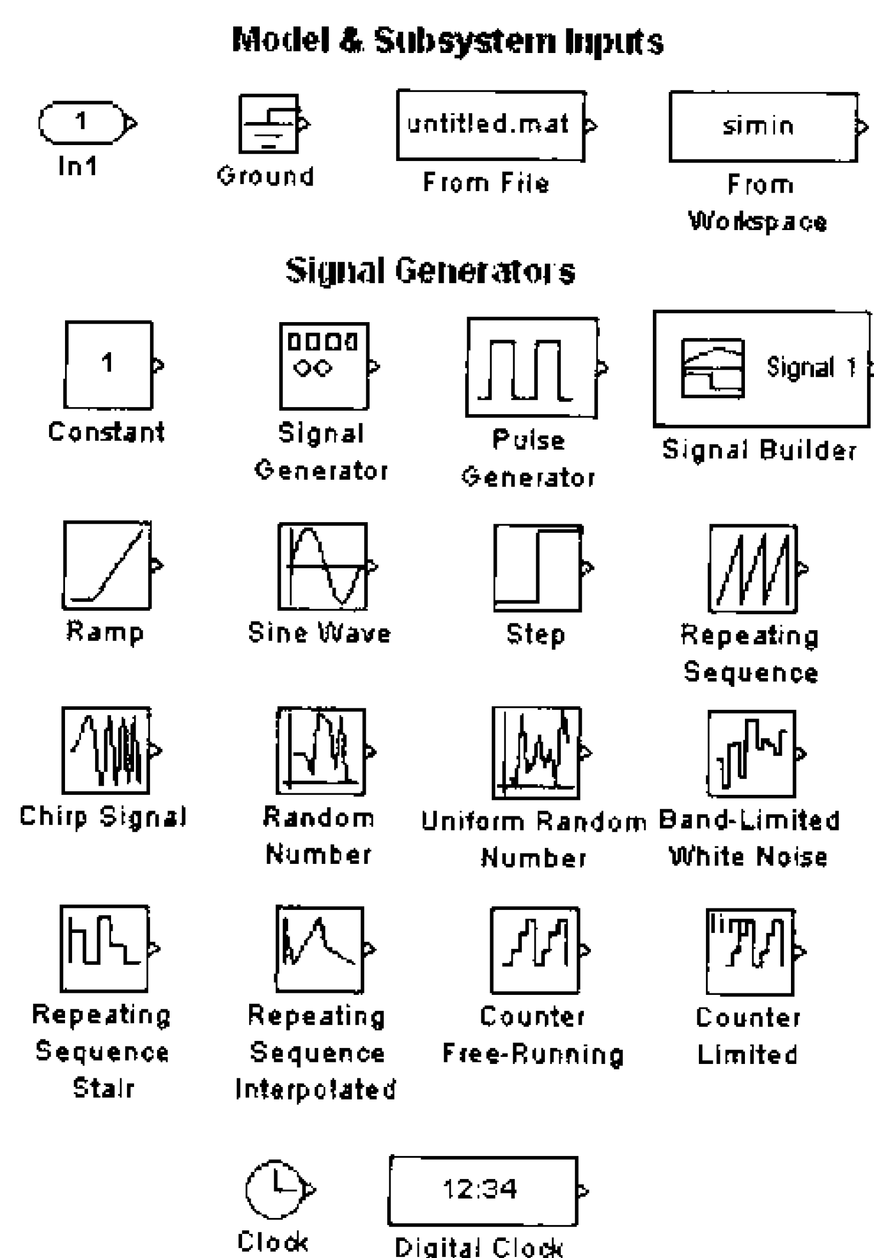


图 6-13 信号源模块组

表 6-11 信号源模块及其功能

名 称	功 能	名 称	功 能
Constant	常数	From File	从文件读数据
Signal Generator	信号发生器	From Workspace	从当前工作空间定义的矩阵读数据
Step	阶跃信号	Random Number	高斯分布的随机信号
Ramp	线性增加或减小的信号	Uiform Random Number	平均分布的随机信号
Sine Wave	正弦波	Band-Limited White Noise	带限白噪声
Repeating Sequence	重复的线性信号，类似锯齿形	In1	输入单元
Pulse Generator	脉冲发生器，和采样时间无关	Ground	接地
Chirp Signal	频率不断变化的正弦信号	Repeating Sequence Stair	重复的线性信号，类似与阶梯状
Clock	输出当前的仿真时间	Repeating Sequence Interpolated	重复的线性信号，以类似内插值的方式替换
Digital Clock	按指定速率输出当前仿真时间		

11. **用户自定义模块组**

用户自定义模块组如图 6-14 所示，其模块及功能如表 6-12 所示。

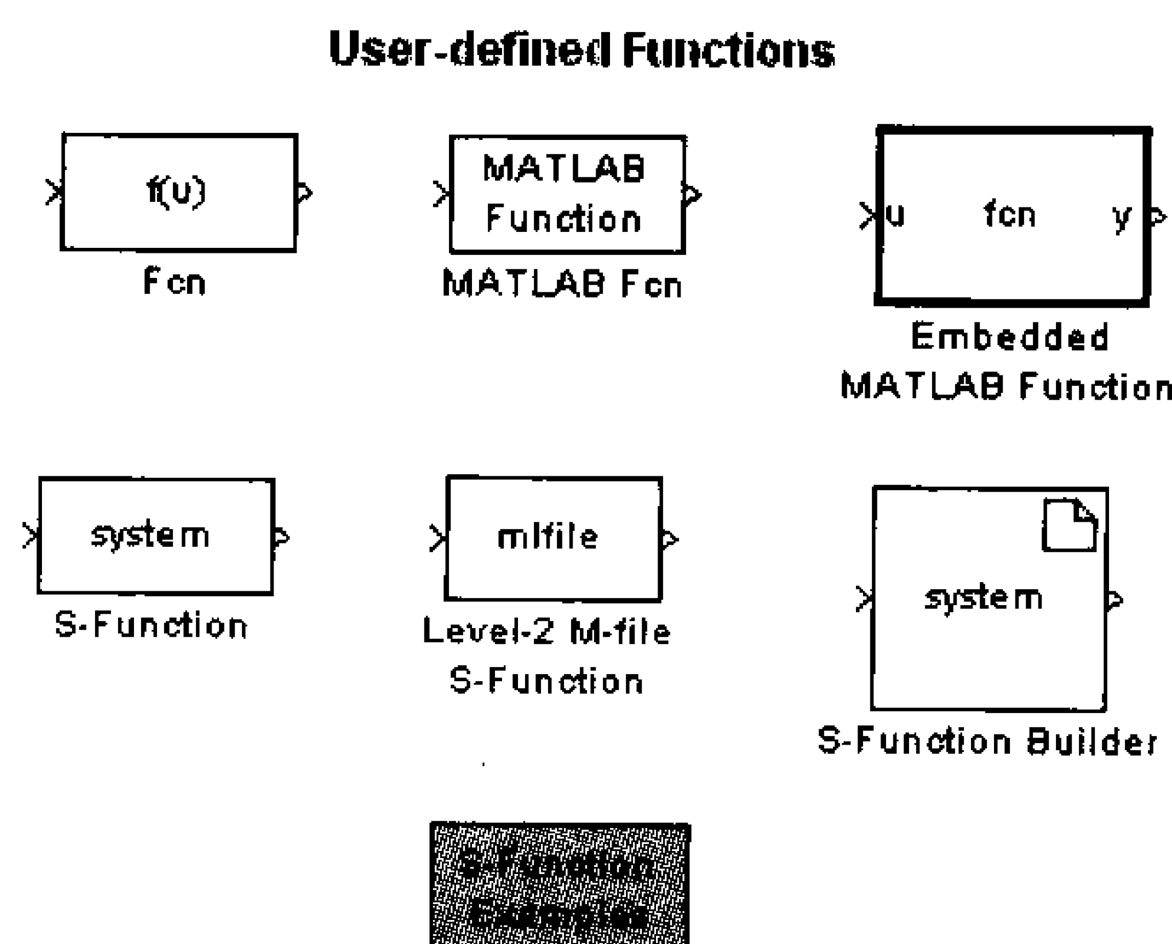


图 6-14　用户自定义模块组

表 6-12　用户自定义模块及其功能

名　　称	功　　能	名　　称	功　　能
Fcn	各种函数组合	MATLAB Fcn	MATLAB 函数
S-Function	S-函数	M-file S-Function	M 文件 S 函数
Embedded MATLAB Function	嵌入式 MATLAB 函数	S-Function Builder	S-函数构造器

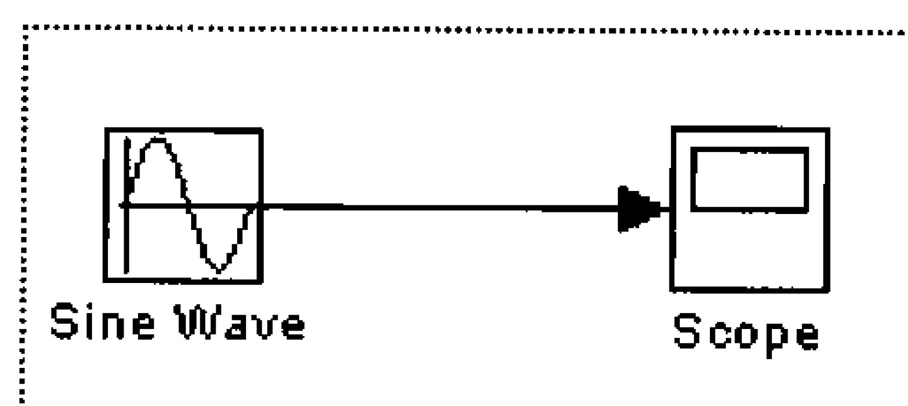
这里我们仅对 Simulink 模块库的组成与基本功能进行初步介绍，它们的详细使用情况请参见本章的后续内容。

6.1.4　Simulink 模块基本操作

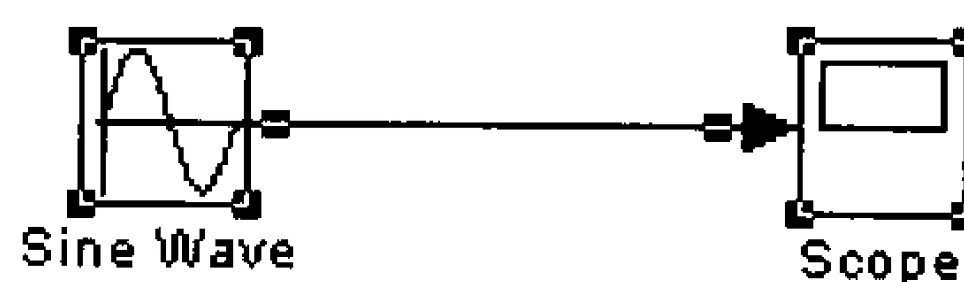
模块作为仿真模型的基本组成单元，其基本操作包括选定、复制、调整、删除、标志和连线等。

1. **模块的选定**

选定单个模块的方法有两种，一种是用鼠标单击欲选择的模块即可，另一种是按下鼠标任意键，拖向该区域的对角，在此过程中会出现虚线框，当虚线框包住了要选的模块（如图 6-15（a）所示）后，放开鼠标键，这时在所有被选模块的角上会出现小黑方块，表示模块（包括与连接模块的信号线）都被选中（如图 6-15（b）所示）。



（a）用虚线框选中单个模块



（b）松开鼠标键后被选中的模块

图 6-15　模块的选定操作

选择活动窗口中所有对象的方法是，在模型窗口中执行【Edit】→【Select All】命令，或者按下【Ctrl+A】组合键即可。

2．模块的复制

1）在同一窗口内复制

有时一个模型需要多个相同的模块，这时复制的方法如下所示。

- 按住鼠标右键，拖动鼠标至合适的地方，释放鼠标。
- 按住【Ctrl】键，再按下鼠标左键，拖动鼠标至合适的地方，释放鼠标。

2）在不同的窗口之间复制

当我们建立模型时，需要从模块库窗口或者已经存在的模型文件窗口把需要的模块复制到新建模型文件的窗口。要对已经存在的模型进行编辑时，有时也需要从模块库窗口或另一个已经存在的模型文件窗口复制模块，有以下两种方法。

- 在一个模型窗口选中模块，按下鼠标左键，将其拖至另一个模型窗口，释放鼠标。
- 在一个模型窗口选中模块，单击“复制”图标，然后用鼠标单击目标模型窗口中需要复制模块的位置，最后用鼠标单击“粘贴”图标即可（此方法适用于任何同一窗口内的复制）。

在如图6-16所示的复制结果中，我们会发现复制出来的模块名称在原名称的基础上又加了编号，这是Simulink的约定：每个模型中的模块和其名称是一一对应的，相同的模块或不同的模块都不能使用同一个名字。复制所得模块具有和源模块一样的属性。我们可以利用复制操作将一个模块插到一个兼容的应用程序中，比如字处理器。



图6-16 模块的复制操作

3．模块的删除

模块的删除，执行【Edit】→【Cut（删除到剪贴板）】或者执行【Edit】→【Clear（彻底删除）】命令，或者在模块上单击鼠标右键，在弹出的快捷菜单中执行【Cut】或者【Clear】命令，也可以在选中模块后，按下键盘上的【Delete】键。

4．模块的调整

1）模块的移动

选中要移动的模块，按下鼠标左键，然后将模块拖到合适的地方即可。但要注意的是，模块移动时，与之相连的连线也随之移动。在不同模型窗口之间移动模块，需要同时按下【Shift】键。

2）模块大小的调整

选中模块，用鼠标选中其周围的4个黑方块中的任意一个开始拖动，这时会出现一个虚线矩形表示新模块的位置，到需要位置后释放鼠标即可。

3）模块的旋转

选定模块，执行【Format】→【Rotate Block】命令使模块旋转 90°，选择【Filp Block】使得模块旋转 180°，效果如图 6-17 所示。

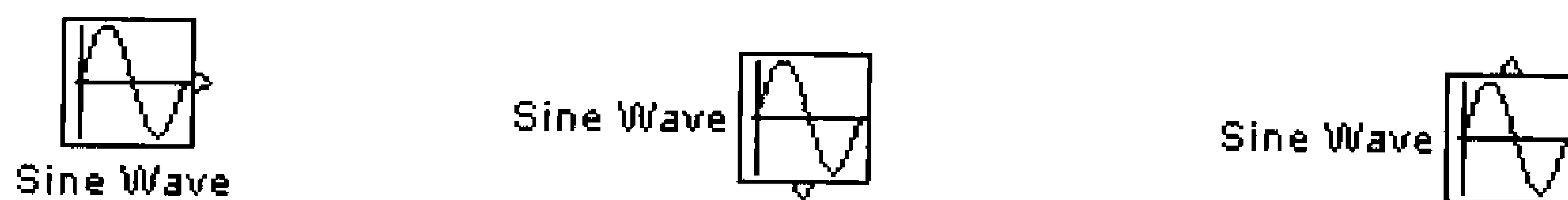


图 6-17　调整模块的方向

4）模块的阴影

选中模块，执行【Format】→【Show Drop Shadow】命令，使模块产生阴影效果，如图 6-18 所示。



图 6-18　模块的阴影效果

5）模块名的处理

①修改模块名：单击模块名，将在原来名字的四周出现一个编辑框。此时，就可以对模块名进行修改了。当修改完毕后，将光标移出该编辑框，单击即可。

②模块名字体设置：选中模块，执行【Format】→【Font】命令，这时会弹出“Set Font”对话框，根据需要设置即可。

③模块名显示与否：执行【Format】→【Hide Name】命令，模块名就会被隐藏，同时 Hide Name 改为 Show Name。选择“Show Name”就会使得模块隐藏的名字显示出来。

④改变模块名的位置：模块名的位置有一定的规律，当模块的接口在左右两侧时，模块名只能位于模块的上下两侧，默认在下侧；当模块的接口在上下两侧时，模块名只能位于模块的左右两侧，默认在左侧。因此，模块名只能从原位置移动到相对的位置。可以用鼠标拖动模块名到其相对的位置；也可以选定模块，执行【Format】→【Flip Name】命令实现相同的移动。

5．模块的参数和特性设置

Simulink 中几乎所有模块的参数（Parameters）都允许用户进行设置。只要双击要设置参数的模块就会弹出设置对话框，如图 6-19 所示。这是正弦波模块的参数设置对话框，可以设置它的频率、幅值、相位和采样时间等参数，模块参数还可以用 set_param 命令修改。

每个模块都有一个内容相同的特性（Properties）设置对话框，如图 6-20 所示。它可以设置模块的优先级、标记和说明等内容。

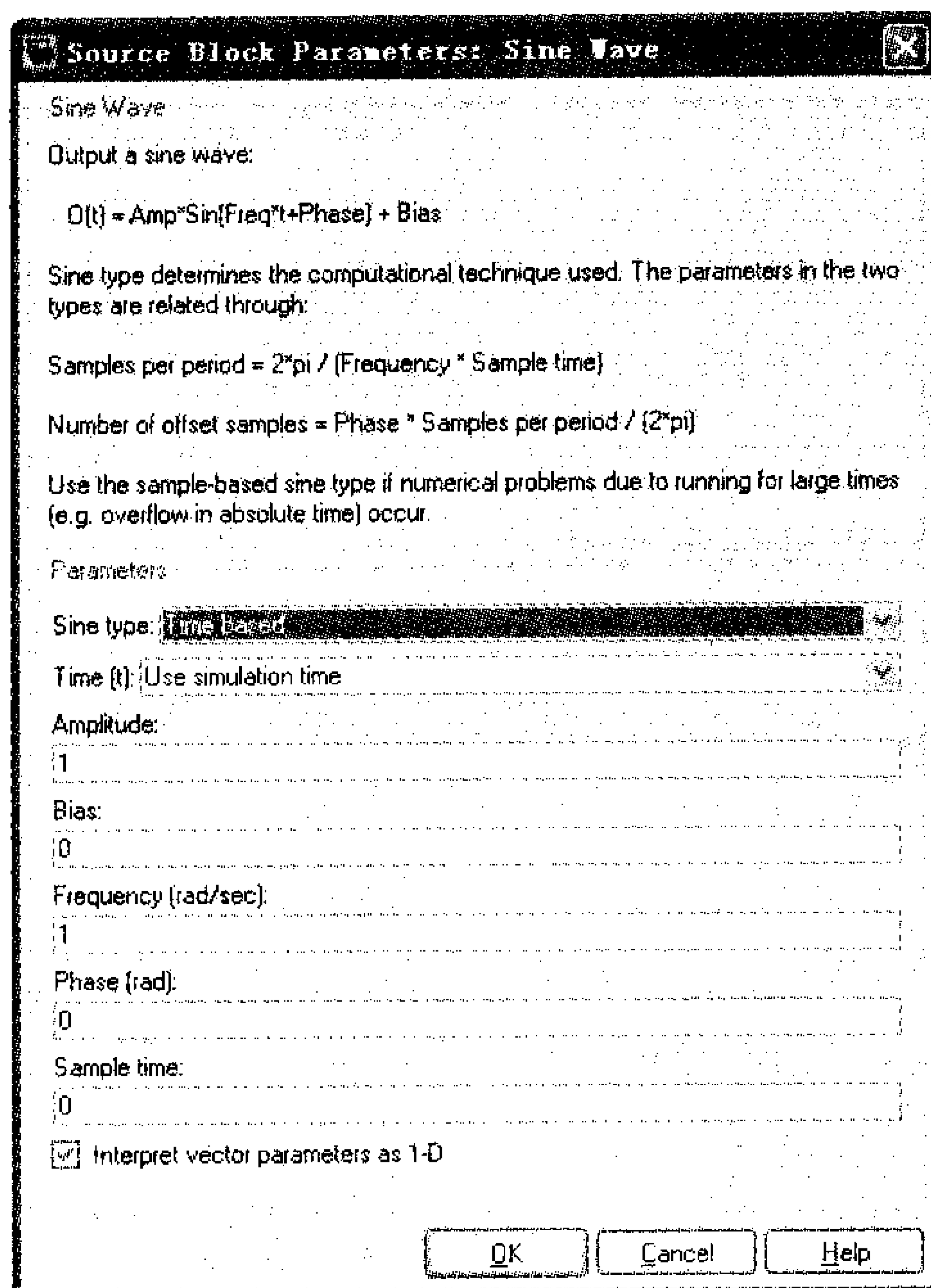


图 6-19 模块参数设置对话框

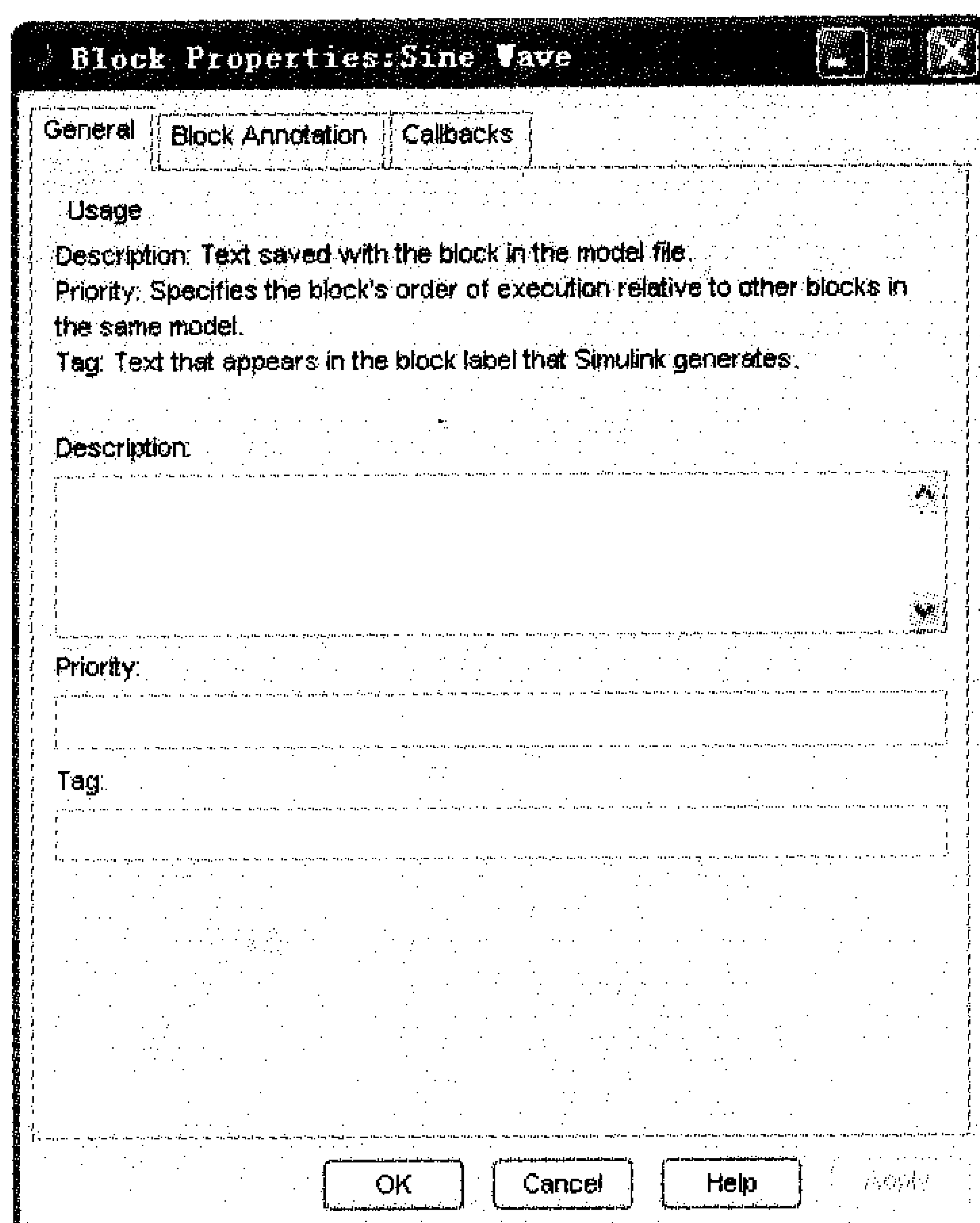


图 6-20 模块特性设置对话框

6．模块的注释

在模型窗口中，书写注释的目的是为了有助于更好地理解模块。注释可以提供模型的文本信息。用户可以在模型的框图中添加文本注释，其方法为：

（1）把鼠标指针移到需要添加文本注释的位置。

（2）单击鼠标左键。

（3）输入注释文本。

（4）再单击鼠标左键。

（5）Simulink 把文本注释放在用户单击鼠标指针所在位置略微偏下一点的地方。

（6）在注释文字处单击鼠标左键，待出现编辑框后，按下鼠标左键，就可以把该编辑框拖曳到任何希望的位置。

7．模块连线

Simulink 模型中的信号总是由模块之间的连线携带并传送，因此模块间的连接被称为信号线（Signal Lines）。当模块设置好后，只有将它们按一定的顺序连接起来才能组成一个完整的系统模型。在连接模块时，要注意模块的输入、输出端和各模块间的信号流向。在 Simulink 中，模块总是由输入口接入信号，由输出口发出信号。

1）连线绘制

这是最基本的情况，需要从一个模块的输出端连接到另一个模块的输入端。其绘制过程如下。

（1）将鼠标指针移到对象的输出或输入点，用户只需要把鼠标移到靠近端口的任何位置，而没有必要非常精确地移到哪个端口上，这时鼠标指针将会变成十字状。

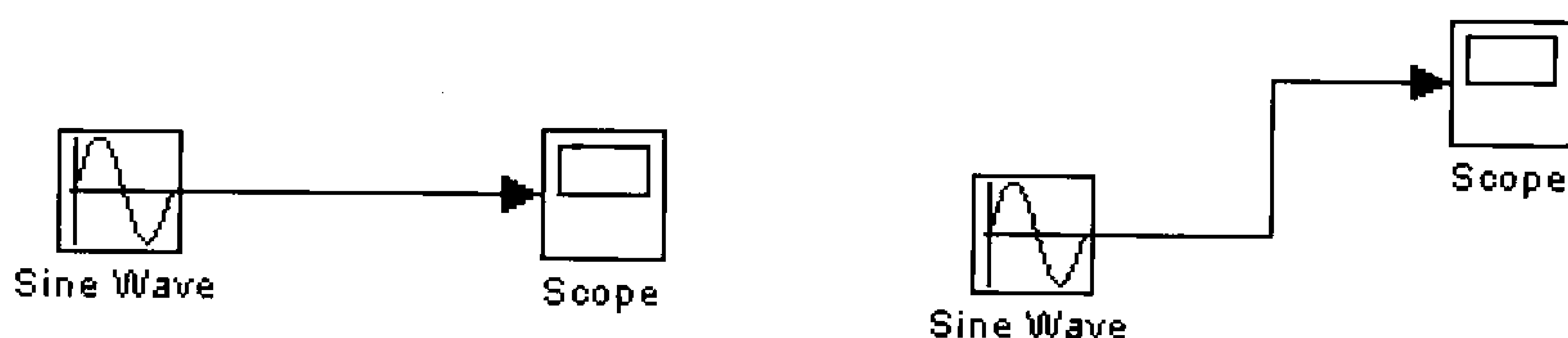
（2）按住鼠标左键，拖曳至另一个对象的输入或输出点，用户只需把鼠标指针移动到靠近模块输入端口的任何位置。如果用户把鼠标指针移到了第二个模块里面，则把连线连到第二个模块中一个没有被占有的输入端口。如果想把连线连到指定的端口，就必须在释放鼠标左键以前把鼠标指针定位到那个输入端口，此时，鼠标指针将变成双十字状。

（3）放下鼠标左键，完成连接，产生连接线，如图 6-21（a）所示。



信号的流向不会因为连线的方向不同而发生改变。

如果两个模块不在同一水平线上，连线将是一条折线，如图 6-21（b）所示。



（a）水平连接　　（b）折线连接

图 6-21　模块连线绘制

2）连线移动和删除

调整模块间连线位置的情况可以采用鼠标的简单拖曳来实现，其操作过程如下。

（1）选中待移动的线段，将光标指向它。

（2）按下鼠标左键，拖至目标地点后，释放鼠标。

（3）此时，就能把鼠标移到目标位置。

过程如图 6-22 所示。

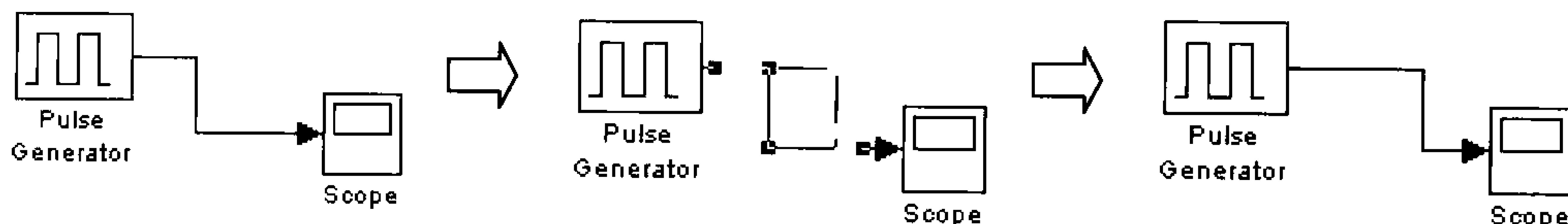


图 6-22　移动连接线段

删除线段时，选中待删除的线段，当四周出现黑色小框时，按下键盘上的【Delete】键，即可删除线段。也可以通过【Edit】菜单或右键菜单的【Delete】指令来删除线段。

3）连线分支

在实际模型中，一个信号往往需要分送到不同模块的多个输入端，此时就需要绘制分支线段（Branch Line）。比如，反馈控制系统中，反馈线的绘制就要使用分支操作产生。

其绘制过程如下。

（1）将鼠标指到要产生分支的线段上。

（2）按住【Ctrl】键，再按下鼠标左键（或按住鼠标右键）。

（3）拖曳鼠标，即可拉出分支线段。

（4）将分支线段拉到另一对象的输入端，如图6-23所示。

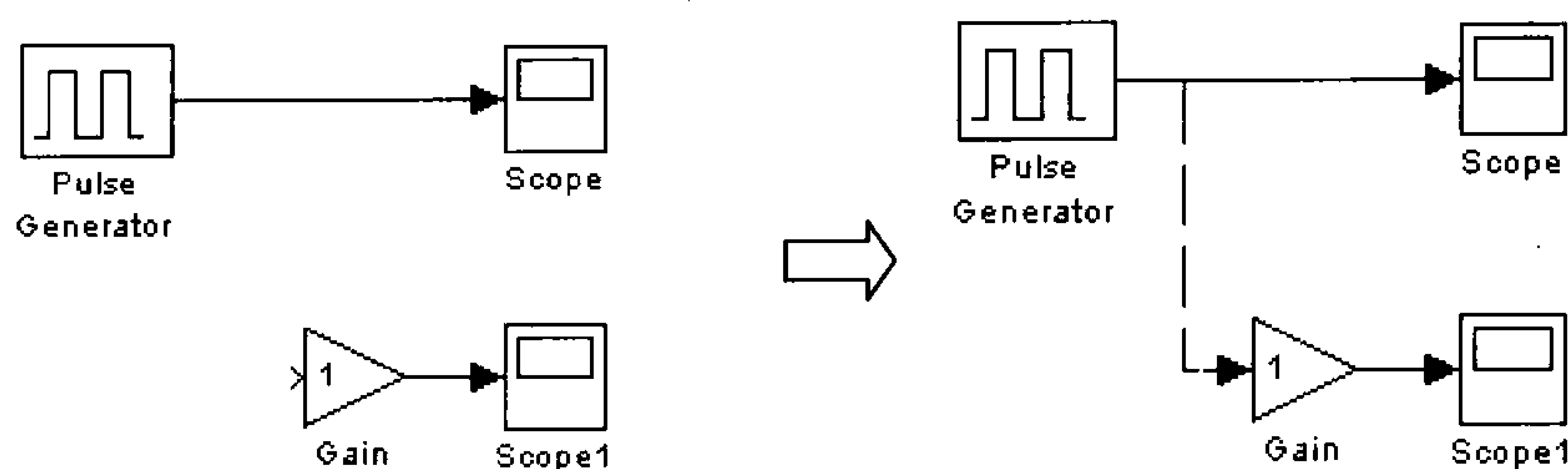


图6-23 拉出分支线段

4）连线的折曲和调整

在构建方块图模型时，有时需要使两模块连续转向，以让出空白来绘制其他事物。产生“折曲”的过程如下。

（1）选择要折曲的线段。

（2）将鼠标指到线段的控制点上，直到箭头变换十字形式为止。

（3）按下鼠标左键，拖曳线段，即可将线段进行分段折曲，如图6-24所示。

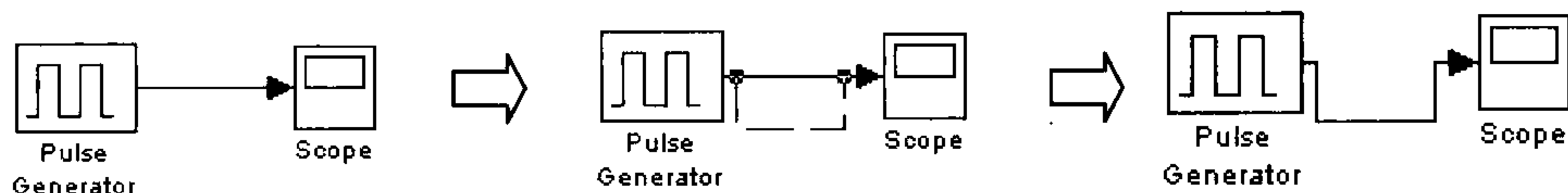


图6-24 信号线的折曲

如果不想以转直角的方式分段，也可以在线段的任一位置，按住【Shift】键和鼠标左键，以任意角度产生折曲，如图6-25所示。

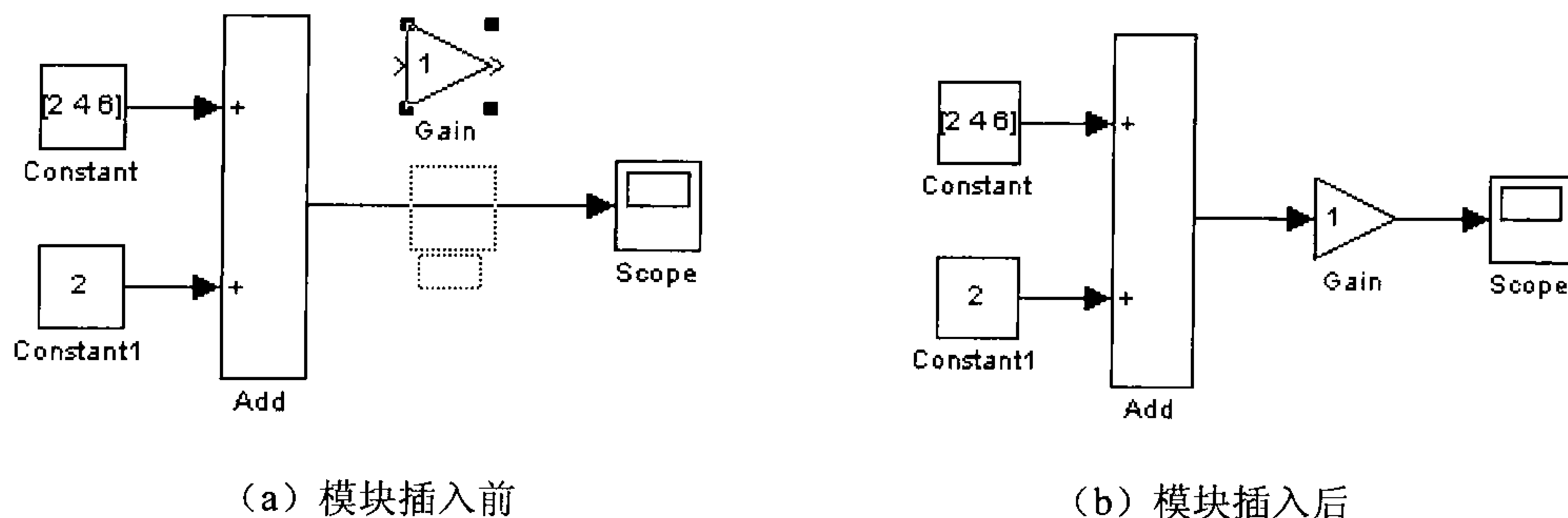


图6-25 任意角度的折曲

移动折点的方法是：选中折线，将光标指向待移的折点处，当光标变成一个小圆圈的时候，按下鼠标左键并拖动鼠标至目标处，释放鼠标。

5）连线中的模块插入

如果模块只有一个输入口和一个输出口，那么可以将该模块直接插到一条信号线中，如图6-26（a）所示。模块插入后的结果如图6-26（b）所示。



（a）模块插入前　　　（b）模块插入后

图 6-26　连线中的模块插入

6）连线数据类型的表示

根据模块传输的数据类型的不同，模块的连线状况也不一样。这样通过模块之间的连线，用户可以了解其传递数据的结构。

①显示数据类型和信号维数。在连线上可以显示数据的类型：执行【Format】→【Port】→【Signal Display】→【Port Data Types】命令，可以在线段上显示出数据的类型。执行【Format】→【Port】→【Signal Display】→【Signal Dimensions】命令，可以在连线上标出输入输出信号的维数，如图 6-27 所示。

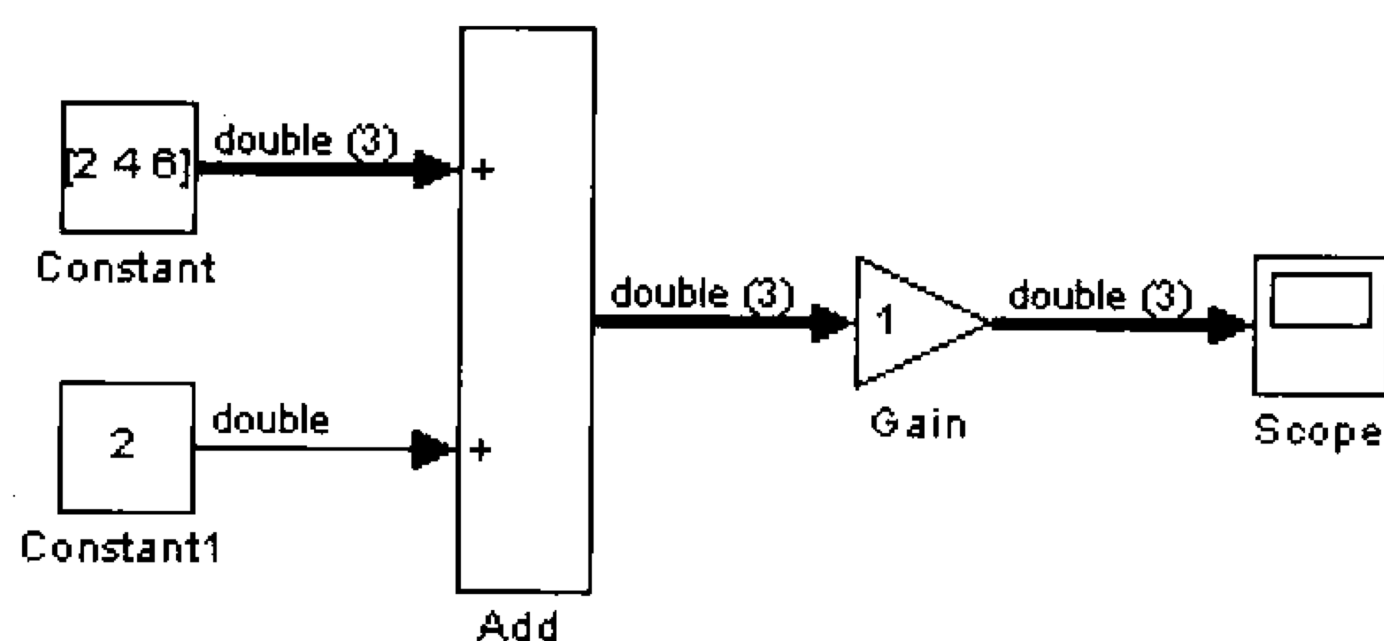


图 6-27　在连线上显示数据类型和信号维数

②用粗线表示向量。几乎所有的模块既能接收标量输入，也能接收向量输入，并允许用户定义标量或者是向量的参数。为了能够比较直观地区别各个模块之间传递的数据是变量还是矩阵（向量），可以通过执行【Format】→【Port】→【Signal Display】→【Wide NonScalar Lines】命令来定义模型中的哪一条线传递的是向量信号，Simulink 就把传递向量信号的线画得比传递标量信号的线宽一些，如图 6-28 所示。如果偶尔出现显示未及时更新的情况，则可以通过执行【Edit】→【Update Diagram】命令更新显示。另外，仿真开始时也进行这样的更新显示。

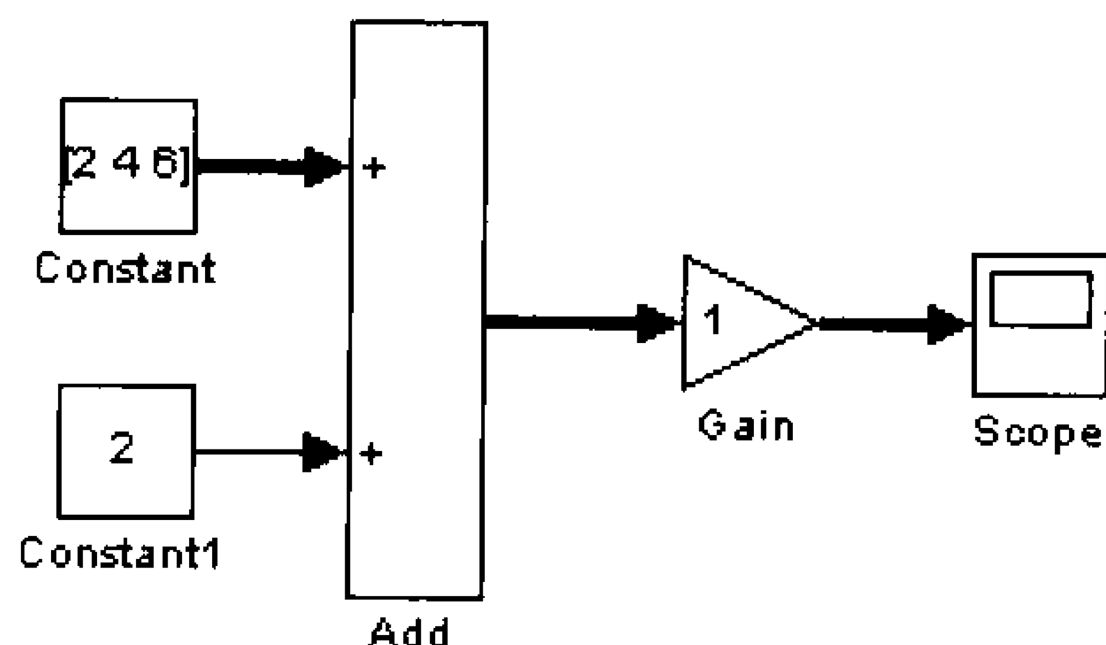


图 6-28　用粗线表示非标量向量

③标量扩展的连线表示。标量扩展是指把一个标量变成一个具有相同元素的向量。Simulink 只对一个模块的输入和参数进行标量扩展。具体哪些模块可以扩展，用户可以在使用时注意模块的说明。

- 输入的标量扩展。当某个模块（如 Sum 和 Relational Operator 等）有一个以上的输入时，用户可以把向量输入和标量输入混合起来。在这种情况下，那个标量输入信号就要进行扩展，形成一个具有和向量输入信号维数一样的具有相同元素的向量。
- 参数的标量扩展。对于可以进行标量扩展的那些模块，其参数既可以定义为标量，又可以定义为向量。当定义为一个向量参数时，向量参数中的每一个元素与输入向量中的每一个元素相对应。而定义为一个标量参数时，Simulink 就对标量参数进行标量扩展，自动形成一个具有相应维数的向量。

7）连线标记

为了使模型更加直观，可读性更强，我们可以为传输的信号连线作标记。

①添加标记。双击需要添加标记的信号线，弹出一个空白的文本框。在其中输入文本，作为对该信号线的标记。输入结束后，只需要将光标移出该编辑框，单击鼠标键即可。

②修改标记。单击需要修改的标记，原标记四周出现了一个编辑框，此时即可开始修改标记。

③移动标记。单击标记，待编辑框出现后，将光标指向编辑框，按下鼠标后拖动至新位置即可。

④复制标记。类似于移动标记，只是要求同时按下【Ctrl】键，或者改用鼠标右键操作。

⑤删除标记。单击标记，待编辑框出现后，双击标记使得整个标记被全部选中，按下【Delete】键。最后，将光标移出编辑框后单击鼠标，就可以删除标记。

⑥信号传递标记。信号标记可以随着信号的传输从一些模块中进行传递。支持这种传递的模块有 Mux、Demux、Inport、From、Selector、Subsystem 和 Enable。要实现信号标记的传递，需要在上面列出的某个模块的输出端建立一个以“<”开头的标记。当开始仿真或者是执行【Edit】→【Update Diagram】命令时，传输过来的信号标记就会显示出来，如图 6-29 所示。

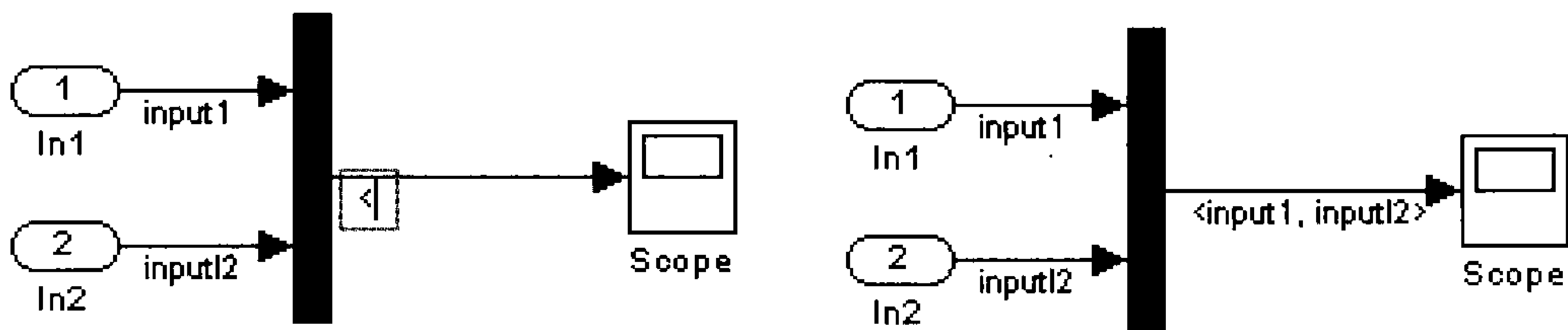


图 6-29　信号线传递的标记

6.1.5 Simulink 仿真参数设置

启动 Simulink 仿真有以下两种形式。

- 当执行【Simulation】→【Start】命令时，Simulink 以一种数值解算方法去求解方程。
- 在 MATLAB 命令窗口中输入仿真指令。

两种方法各有所长，第一种方法比较常用，运行【Start】后系统开始仿真。此时，【Start】变成了【Pause】，选择【Pause】可以暂停执行仿真，如果要终止则选择【Stop】命令。用户通过菜单运行仿真，可以直接观察系统的运行结果。不用记住那些命令格式，只要利用简单的窗口界面，就可以实时修改模型，也可以同时仿真多个模型。相比较而言，这是最方便的方法，适合初期系统分析阶段使用。

在进行仿真前，如果用户不采用默认设置，那么就必须对各个仿真参数进行配置，执行【Simuliation】→【Configuration Parameters】命令。采用菜单方式进行仿真最主要的就是设置参数，其中包括：仿真的起始和终止时刻的设置；仿真步长的选择；各种仿真容差的选定；数值积分算法的选择；是否从外界获得数据；是否向外界输出数据等。

通过对树形菜单的选择，打开对话框对参数进行设置，得到的树形结构如图 6-30 所示。

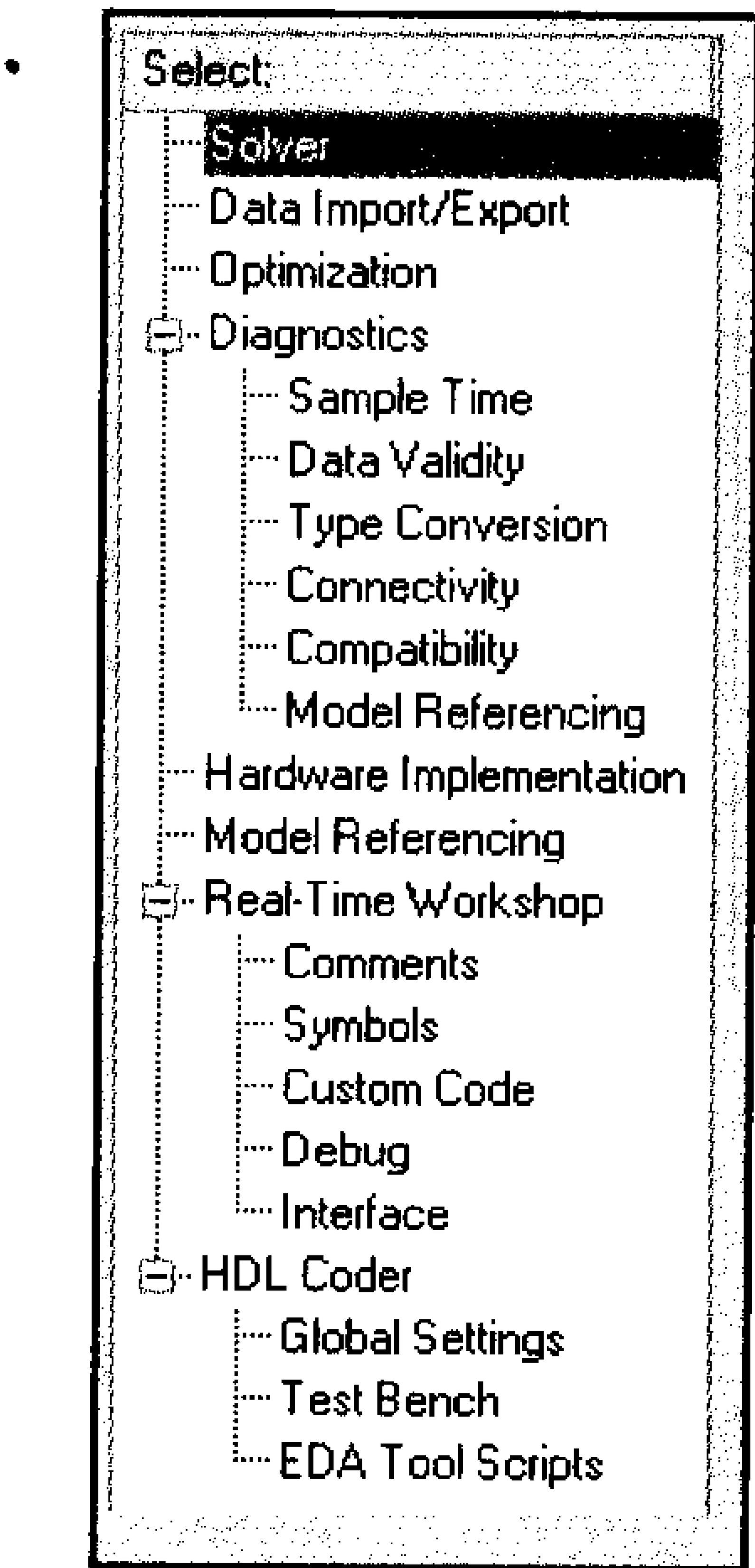


图 6-30　仿真参数设置

1．解算器（Solver）设置

在 Solver 页里需要设置仿真的起始和截止时间，选择合适的解法（Solver）并指定参数。对于不同的模型和不同的条件，仿真的性能（如仿真的速度、精度等）是千变万化的。所以，在选择仿真方法时，必须时刻牢记这些差异。Solver 页设置对话框如图 6-31 所示。

图 6-31　Slover 参数设置对话框

1）仿真时间设置

设置起止时间就是在“Start time”和“Stop time”的文本框中输入相应的数值，其默认值分别为“0”和“10”，单位为“秒”。但是由于运行时间和计算机性能、模型复杂度、解法、步长等很多因素有关，其最后实际所用的时间和设置的时间有点出入。

仿真的主要过程一般是求解常微分方程组。“Solver options”的内容主要是针对解常微分方程的。“Type”栏用来设置解算器类别。解算器分为两大类别：变步长（Variable-step）解算器和定步长（Fixed-step）解算器。“Solver”栏设置解算器的具体算法类型，如 ode45、ode23、ode113 和 ode15s 等。默认设置是选择变步长的 ode45。这种解算器能在保证精度时使用尽可能大的步长，完全排除积分步长和输出“解点”间隔之间的相互制约，可不必为获得光滑输出而设置很小的步长。

2）变步长解算器的步长和容差设置

变步长解法可以在仿真过程中根据要求调整运算步长，在采用变步长解法时，应该先指定一个容许误差限（在“Relative tolerance”或者“Absolute tolerance”中设置），使得当误差超过这个误差限时自动修正仿真步长。所以，在变步长仿真时误差限的设置将关系到常微分方程组解的精度。同时，采用变步长解法还要设置最大步长和最小步长。在默认情况下，系统将按照下面的公式决定最大步长。

最大步长=（停止时间–起始时间）/50

可以在“Initial step size”栏中设置初始步长，在“Zero crossing control”栏中设置零交叉控制。

2．仿真数据输入输出设置

Simulink 模型可以看做一组联立的一阶微分或差分方程。构成模型的传递函数模块、状态方程模块、（某些）非线性模块等都伴随着相应的状态变量，于是就引出状态变量的存取问题，解决存取问题最简单的途径是利用输入输出设置页。

如图 6-32 所示，在这一部分可以设置 Simulink 和当前工作区的数据输入输出。通过设置，可以从工作区输入数据、初始化状态模块（State），也可以把仿真结果、状态模块数据、时间数据保存到当前工作区。输入输出的数据格式可以是矩阵和结构（Structure）数组等。

Load from workspace
Input: [t, u]
Initial state: xInitial
Save to workspace
Time: tout
States: xout
Output: yout
Final states: xFinal
Signal logging: logsout
Inspect signal logs when simulation is paused/stopped
Save options
Limit data points to last: 1000 Decimation: 1
Format: Array
Output options: Refine output Refine factor: 1

图 6-32　Data Import/Export 页面设置

1）Load from workspace 区

①Input 栏：假如模型窗口中使用了输入模块 In，那么就必须勾选“Input”复选项，并填写在 MATLAB 工作空间中的输入数据变量名，比如[t,u]或者是 TU。倘若输入模块有 n 个，则 u 的第 1，2，…，n 列分别送往输入模块 In1，In2，…，Inn。输入的格式可以是矩阵、包含时间数据的结构或者是一般数组结构。各种结构都有比较严格的要求。矩阵的列数要求等于模型输入端口的个数加 1，Simulink 自动地把第 1 列当成时间向量，后面几列按顺序依次对应各个端口。

对于包含时间数据的结构，Simulink 有更加严格的规定。首先结构必须有两个顶级域：times 和 signals（域名不能改变）。顶级域 time 包含一个列向量，表示仿真时间。顶级域 signals 包含一个数组，数字的每个元素都是一个子结构，每个子结构对应模型的一个输入端口，每个子结构下必须包含域 values，values 域包含一个列向量，就是与该子结构相对应的输入端口的输入数据。对结构的赋值应该分别按照结构的各个域进行。

②Initial state 栏：该复选框的勾选，将强迫模型从工作空间中获取模型所有状态变量的初始值，而不管构建该模型的“积分块”是否设置过什么样的初始值。该栏空白处填写的变量名（默认名为 xInitial）应该是工作空间中存在的变量。该变量包含着模型状态向量的“初始值”。

2）Save to workspace 区

在参数设置的“Save to workspace”区域，可供选择的项有时间（Time）、端口输出（Output）、状态（States）和最终状态（Final State）。勾选选项前面的复选框并在选项后面的文本框输入变量名，就会保存相应的数据到指定的变量。

①Time 栏：勾选该复选框后，模型将把（时间）独立变量以指定的变量名（默认名为 tout）存放于工作空间。选取不同的数据格式，Time 的含义是不变的，总是仿真的采样时间。

②States 栏：勾选该复选框，模型将把其状态变量以指定的变量名（默认名为 xout）存放于工作空间。使用矩阵格式时，States 输出的矩阵每一列对应于一个模型的状态模块。

③Output 栏：假如模型窗口中使用输出模块 Out，那么就必须勾选“Output”复选框，并填写在 MATLAB 工作空间中的输出数据变量名。数据存放和数据输入的情况相似。

④Final states 栏：勾选该复选框，将向工作空间以指定的名称（默认值为 xFinal）存放最终状态值。若该最终状态向量在该模型的新一轮仿真中又被用作初值，那么这一轮仿真是前一轮仿真的“继续”。

使用常用的矩阵格式时，States 输出的矩阵每一列对应于一个模型的状态模块，每一行对应于一个确定时刻的状态；Final State 输出的矩阵只能是一行，表示每个模型的最终状态，每一列对应于一个模型的状态模块；output 输出的矩阵每一列对应于一个模型的状态模块的输出端口，每一行对应于一个确定时刻的输出。



Output 输出的矩阵是到输出端口的值，因此模型中需要有 outport 模块，否则不能生成输出矩阵。

⑤Signal logging name 栏：Simulink 在对某个模型进行仿真的同时创建一个类的实例，用来包含创建过程中所产生的信号记录。该信号记录的名字可以自己定义，其默认值为 logsout。

3）Save options 区

在“Save options”选项组中，用户可以指定输出数据的限制和格式，该区的使用必须和 Save to workspace 配合。

①Limit data points to last 栏：勾选该复选框后，可设置保存变量接收数据的长度，默认值为 1000。假如输入数据长度超过设置值，那么最早的“历史记录”将被清除。

②Decimation 栏：设置“解点”保存频度，默认值为 1。

③Format 栏：保存到工作区的数据也有矩阵、包含时间数据的结构和一般结构这几种格式，在“Save options”区域中的 Format 下可以选择需要的格式。在 Simulink 6.6 中，Format 格式提供了 3 种格式：Array、Structure 和 Structure with time。

④Output options 栏：对同样的信号，选择不同的输出选项后，到输出设备上的信号是不完全一样的。要根据不同的需要选择不同的输出选项，以达到满意的输出效果。其输出选项包括 3 种：Refine output（细化输出）、produce additional output（产生附加输出）和 produce specified output only（只产生指定输出）。

细化输出，可以增加输出数据的点数，使得数据波形更加平滑。

产生附加输出，允许指定产生输出的附加时刻。选择该项后，会出现输出时刻文本框，在这里可以输入计算附加时刻的表达式或者附加时间向量。和细化输出不同的是，这种方式改变仿真步长以使它和指定的附加输出的时间相一致。

只产生指定输出，只在指定输出的时刻产生仿真输出。这种方式改变仿真步长，以使它和指定的产生输出的时刻相一致。

3. 诊断参数设置

在该设置中，用户可以指定在仿真中遇到各种情况时所需要的系统操作。关于算法诊断的设置对话框如图 6-33 所示。

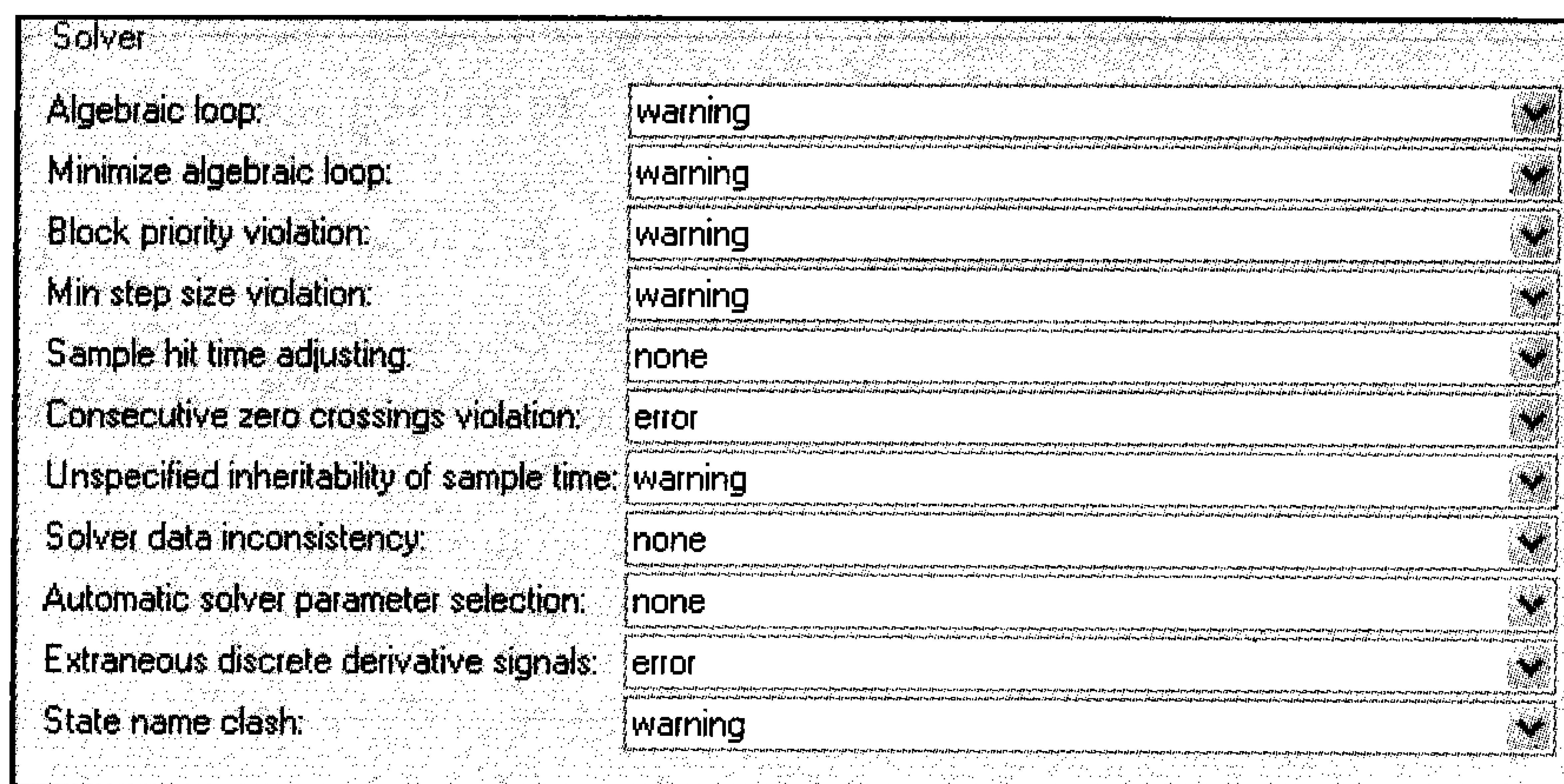


图 6-33 算法诊断设置对话框

从上图可以看出，在对话框中对选择算法可能遇到的各种问题都给出了系统反应的设置。对仿真过程中出现的问题的反应有 3 种类型："None（不做反应）"、"Warning（警告）"和 "Error（提示错误）"。警告信息不影响程序的执行，而提示错误后程序就会停止运行。用户可以根据自己的调试需要作出相应的选择。

4. 实时工作间设置

"Real-time Workshop"（实时工作间）是一种可以在多平台上运行的产品。作为 Simulink 的一个重要功能模块，它是一种实时开发环境，可以直接从 Simulink 的模型产生出可移植的程序源代码，并自动构造出能在多个环境中实时执行的程序。它为系统从设计到实现提供了一条快捷的途径，而且简单易用。通过 Real-time Workshop 可以在远程处理器上运行仿真模型，也可以在主机或外部计算机上运行高速单机仿真。

它主要应用于以下几个方面。

- 实时控制。
- 实时信号处理。先用 MATLAB 和 Simulink 设计信号处理的算法，然后生成程序源代码，编译后移植到硬件上。
- 交互式实时参数调整。可以把 Simulink 作为实时模型的前端，这样就能够在程序执行过程中调整参数。
- 实时仿真。比如模拟系统的训练，实时模型检验和测试。
- 高速单机仿真。
- 产生可移植的 C 语言代码。

Real-time Workshop 的详细介绍可参见本章"Simulink 模型的实时代码生成技术"一节的内容。

6.1.6 Simulink 模型保存与打印

为了方便模型的使用，在关闭模型时，用户可以对模型进行保存。同时在模型窗口中可以直接打印模型，这对复杂模型的书面保存显得尤为重要。

1．保存模型

用户可以通过执行【File】→【Save】命令或者【Save As】命令来保存模型。保存对话框如图 6-34 所示。

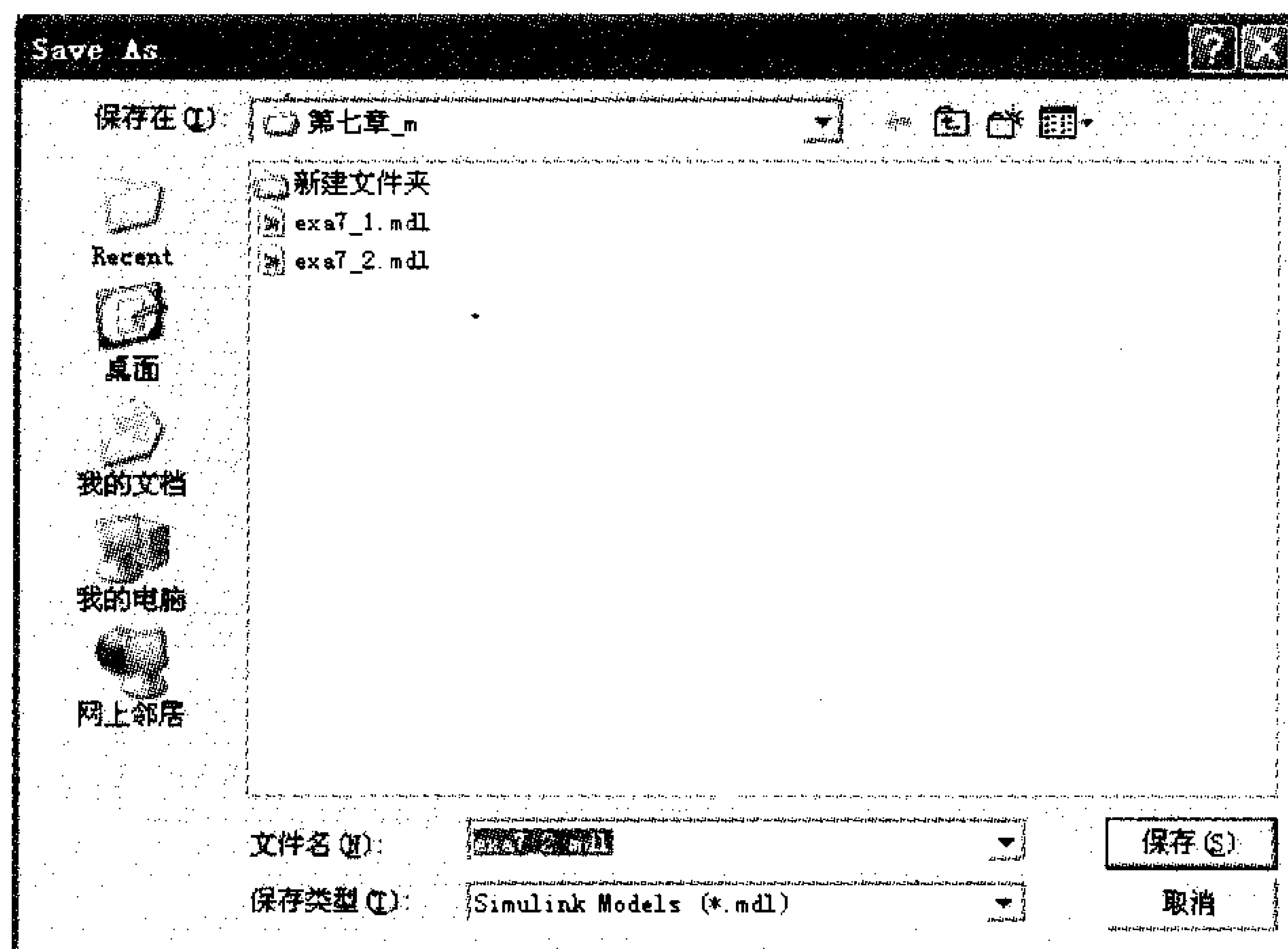


图 6-34 “保存”对话框

保存模型后，Simulink 自动生成一个特殊类型的文件，称为模型文件（Model File），以.mdl 为后缀名，这个文件包括模块的图示和模块的属性。如果用户第一次保存模型，使用【Save】命令提供文件名和保存的路径。模块的名称由字母、数字和下画线组成，但必须以字母开始，并且不能超过 31 字符。如果用户的文件名已经保存过，使用【Save】命令替换原有的文件，或者用【Save as】命令以新的名称和路径保存文件，其保存步骤如下所示。

（1）如果.mdl 文件已经存在，系统将其重命名为一个临时文件。

（2）Simulink 系统执行所有模块的 PreSaveFcn 回调函数过程，然后执行模块图示的 PreSaveFcn 回调函数过程。

（3）Simulink 系统向模型文件写入新的文件并且加上扩展名.mdl。

（4）Simulink 系统执行所有模块的 PostSaveFcn 回调过程，然后执行模块图示的 PostSaveFcn 回调过程。

（5）Simulink 系统删除临时文件。

如果以上过程产生任何错误，Simulink 系统将临时文件写回原来的文件名，将现在的文件写入一个以.err 为后缀的文件，并且将错误信息写入文件中。

2．打印模型

Simulink 有两种打印模型的方法，一种方法是执行【File】→【Print】命令；另一种方法是在 MATLAB 的命令窗口中输入“Print”命令。

执行【File】→【Print】命令，系统将弹出一个对话框，如图 6-35 所示。

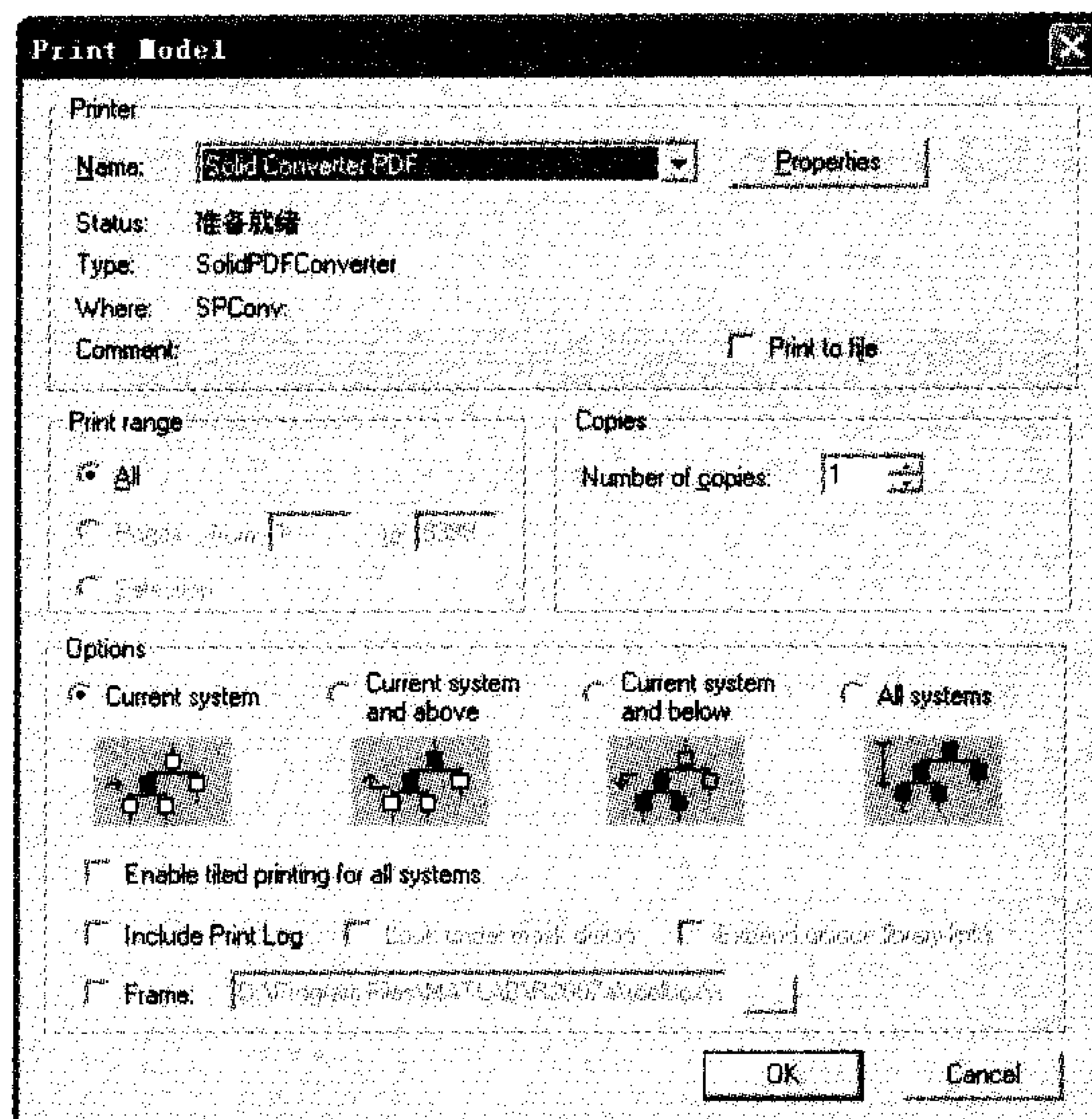


图 6-35 “打印模型”对话框

用户可以在模型中选择性地打印系统，可以有如下几种选择。

- 仅仅打印当前选择的系统。
- 打印当前的系统和所有模型层次上高于它的系统。
- 打印当前的系统和所有模型层次上低于它的系统，可以通过选项确定所封装模块和库模块的具体内容。
- 打印模型中的所有系统，可以通过选项确定所需要封装的模块和库模块的具体内容。

具体的细节设置可以通过执行【File】→【Print Details】命令和【File】→【Print Setup】命令执行。

在 MATLAB 的命令窗口中用“Print”命令打印指定的文件到打印机或者是文件中。但是，这个命令不能打印任何打开的 Scope 模块。“Print” 命令的格式如下：

```
Print-smodel-device filename
```

其中：

- smodel 表示要打印的文件名字，如果省略，则打印当前系统，这时系统必须打开。
- device 表示 Windows 设备，包括：win，当前安装的打印机，以单色打印；winc，当前安装的打印机，以彩色打印；meta，剪贴板，以 M 文件形式打印；bitmap，剪贴板，以位图形式打印；setup，打开“Print setup”对话框，但不打印。
- filename 表示保存输出的文件名。如果指定文件名，则把输出写到指定的文件中。如果文件已经存在了，则改写这个文件。如果指定的文件不包括扩展名，则添加一个适当的扩展名。如果直接把文件打印输出到打印机上，那么就不能控制其大小。如果要控制其打印输出的大小，最好把打印输出先送到一个位图中，然后用 Word 程序来改变其大小。

6.2 Simulink 模型调试

Simulink 作为高性能的系统设计、仿真与分析平台，为用户提供了一个功能强大、界面友好的模型调试工具——Simulink 调试器。通过 Simulink 调试器，用户可以对动态系统模型进行调试，一步一步地进行仿真，发现其中可能存在的问题，并对其进行修改，从而完成系统设计、仿真与分析的目的。

Simulink 的调试方法分为图形界面调试和命令行调试两种。

- 图形界面调试简单方便，可以完成绝大多数情况下用户所需的调试任务。
- 命令行调试的方法是用户输入调试命令来对模型进行调试。可以使用户随心所欲地显示调试中的任何信息，包括图形界面中无法显示的信息，对于一个 Simulink 的高级用户来说，使用命令行调试是很重要的。

本章将分别介绍 Simulink 模型的图形界面调试和命令行调试两种方法，使读者掌握模型的调试技能。

6.2.1 图形调试器基础

在使用图形调试方法时，首先必须熟悉图形调试器，掌握其基本知识。因此这里将介绍图形调试器的启动和功能等知识。

1. 图形调试器启动

Simulink 的图形调试器可以通过执行【Tools】→【Simulink debugger】命令来启动，也可以通过单击常用工具栏中的“调试(Debug)”图标来启动。启动后的调试器如图 6-36 所示。

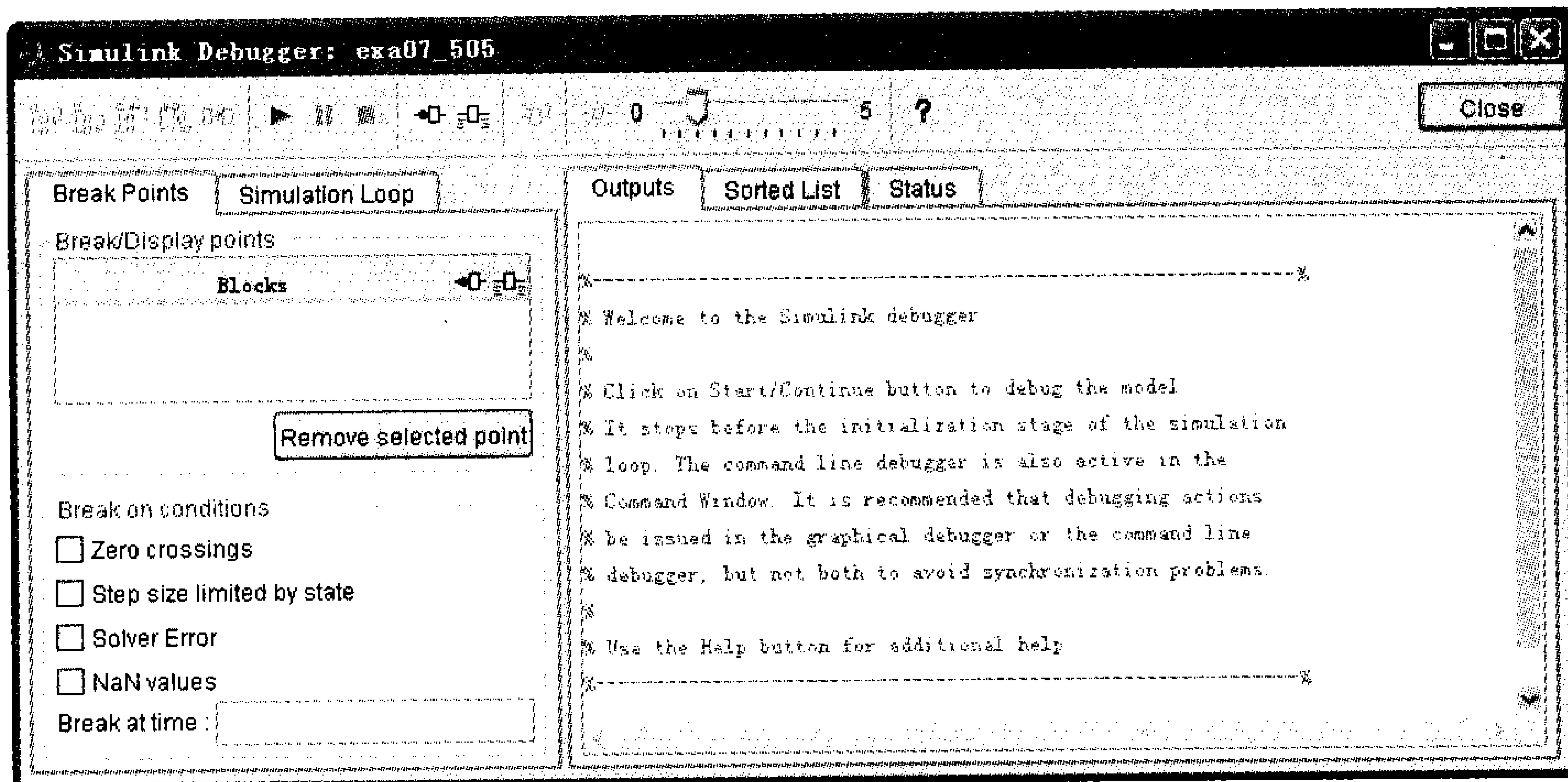


图 6-36 Simulink 模型的图形调试器

2. 图形调试器功能

Simulink 图形调试器具有简单明了的图形界面。这里主要对其调试工具栏、断点条件

设置和显示，以及调试信息输出等功能进行介绍。

1）调试工具栏

图形调试器的工具栏如图 6-37 所示，它位于调试器的上方。

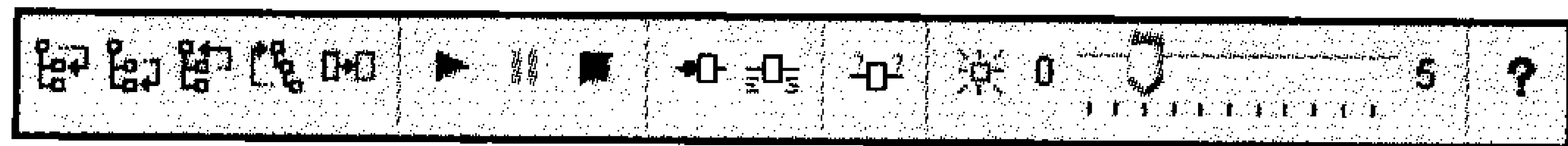


图 6-37 Simulink 调试器工具栏

工具栏中共有 14 个按钮，可以用它们来完成大部分的调试工作，从左到右它们分别执行的功能如下：“进入当前方法”、“跨过当前方法”、“暂时离开当前方法”、“在下一个时间步长返回第一个方法”、“执行下一个模块”、“开始或者继续运行仿真”、“暂停调试过程”、“终止调试过程”、“在选中的模块前设置断点”、“执行时显示选中模块的 I/O 信息”、“显示选中模块的当前 I/O 信息”、“开启或者关闭动画”、“动画延迟”，以及“模块帮助信息”。

2）断点的条件设置和信息显示

用户可以直接在选中的模块前设置普通断点，也可以设置模块条件断点。条件断点是指该模块处在仿真过程中满足一定的条件时才成为断点。Simulink 图形调试器提供了 5 种条件断点设置，如图 6-38 所示。

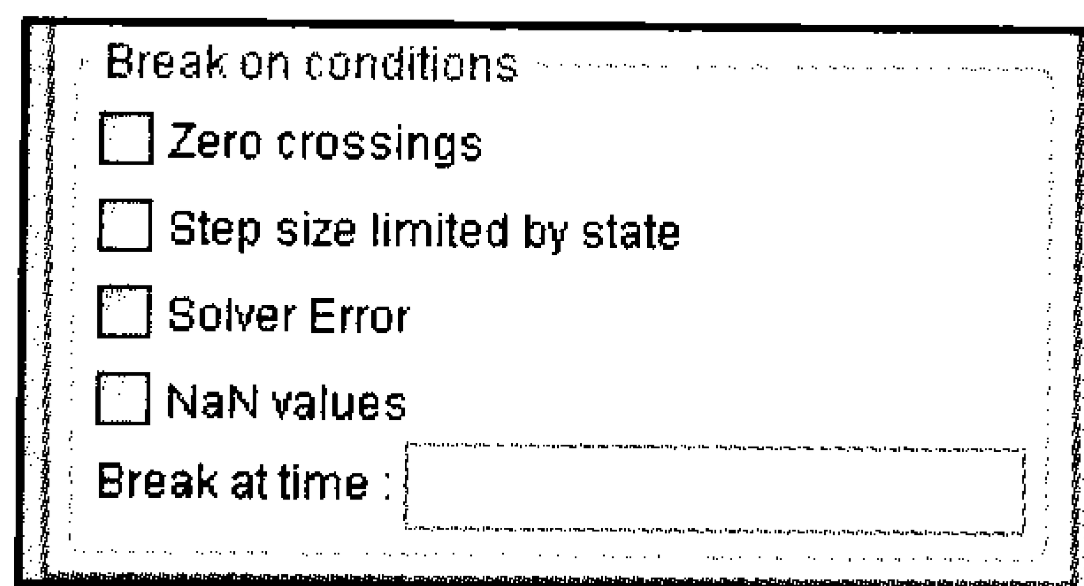


图 6-38 条件断点的设置

据图可知，Simulink 模块有 5 种条件断点设置方法，从上至下分别是：“在系统发生过零时设置断点”、“在仿真步长受到状态限制处设置断点”、“在解算器出现错误处设置断点”、“在系统中出现无穷大值（NaN）处设置断点”和“在制定的仿真时刻处设置断点”。

调试器提供了一个断点信息显示框，如图 6-39 所示。它能够显示当前系统的断点模块，并能够设置是否显示该断点的输入/输出。通过【Remove selected point】按钮可以取消框中被选中的断点。

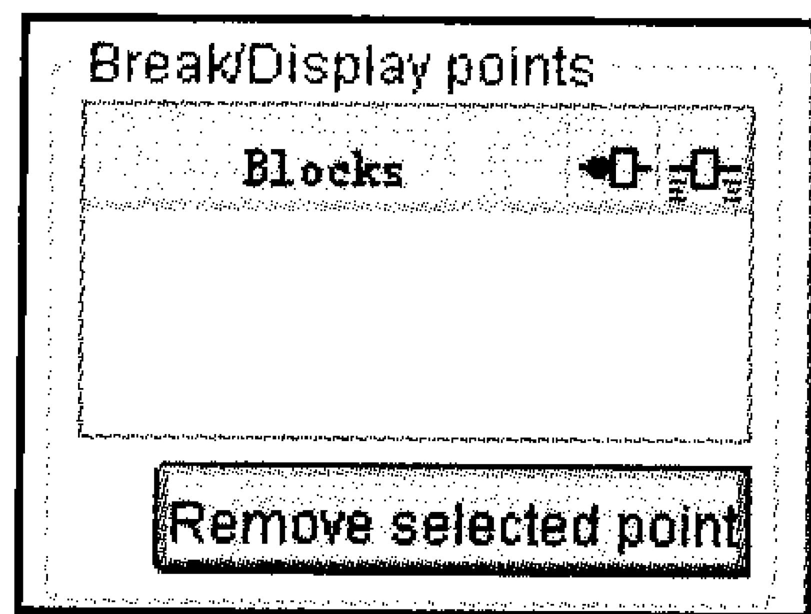


图 6-39 断点信息显示框

3）调试信息输出

在对指定的系统模型进行调试时，调试结果和中间信息均在 Simulink 的输出窗口显示，如图 6-40 所示。该窗口有 3 个选项卡：“Outputs”、“Sorted List” 和 “Status”，它们的功能如下。

- Outputs：输出调试信息和结果，如调试时刻、调试模块和该模块的输入/输出。
- Sorted List：输出调试顺序，即调试过程中的各模块的执行顺序。
- status：输出调试状态，如当前仿真时间、调试命令，以及调试断点等状态信息。

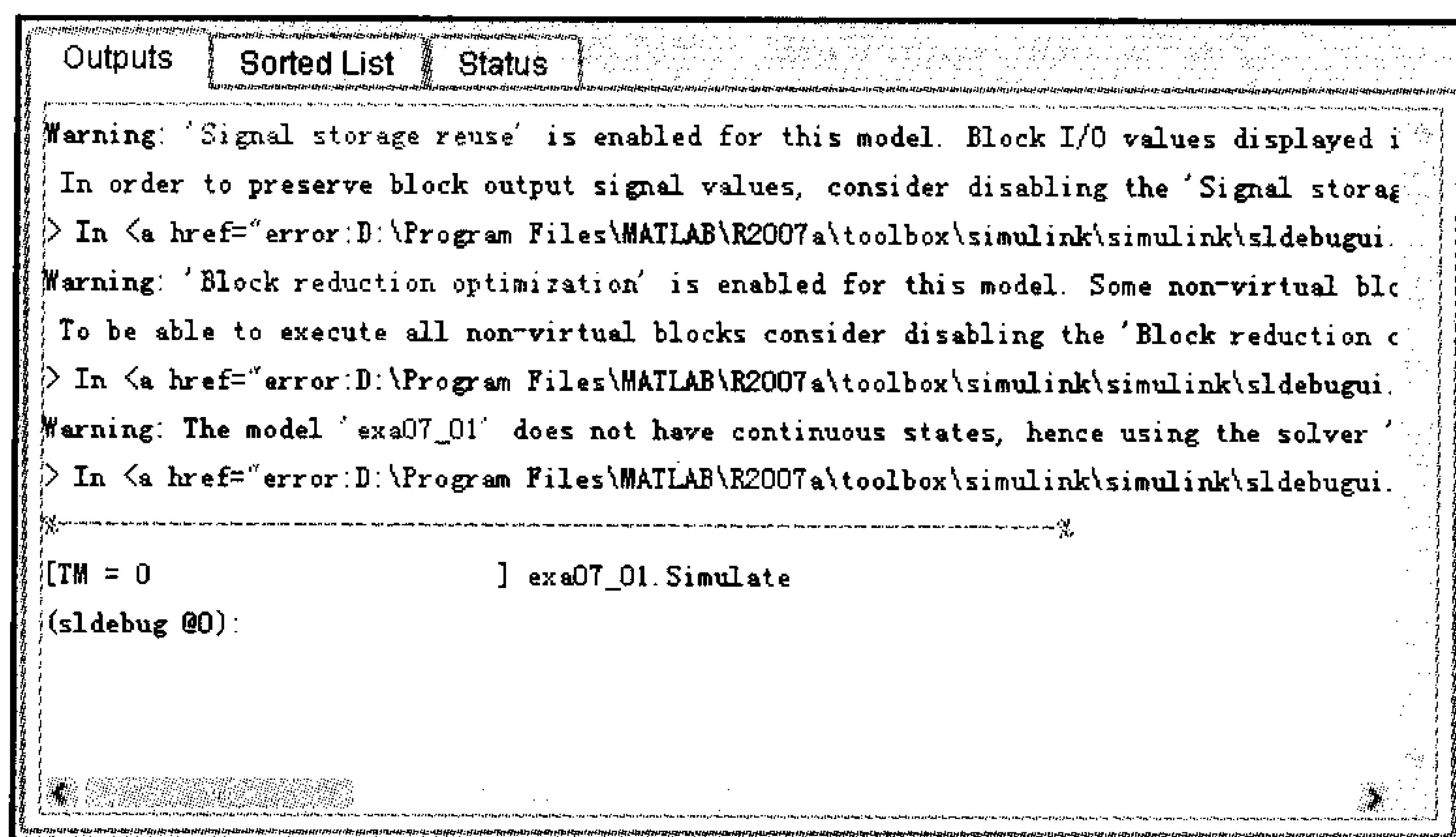


图 6-40 调试信息输出窗口

6.2.2 使用图形调试器调试模型

本小节通过 Simulink 模型实例讲解，让读者掌握使用 Simulink 图形调试器调试模型的一般过程和方法。

例 6.1 构建一个模型系统，使其在一帧二进制序列的末尾附加 16 位的 CRC 校验码，并通过图形调试器对模型进行调试。

依据例 6.1 的要求，构建的 Simulink 模型如图 6-41 所示。

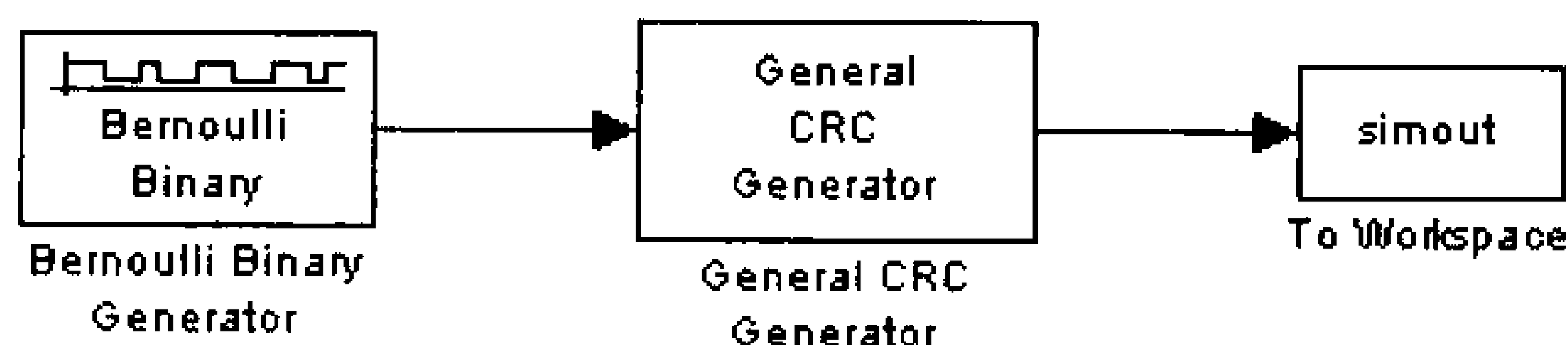


图 6-41 Simulink 调试模型

该模型在伯努利产生器产生的一帧二进制序列的末尾附加 16 位的 CRC 校验码，并将输出结果保存到工作空间中的 simout 变量中。在 3 个模块当中，我们只改变 Bernoulli Binary Generator 模块参数设置，如图 6-42 所示，其他两个模块采样默认参数设置。

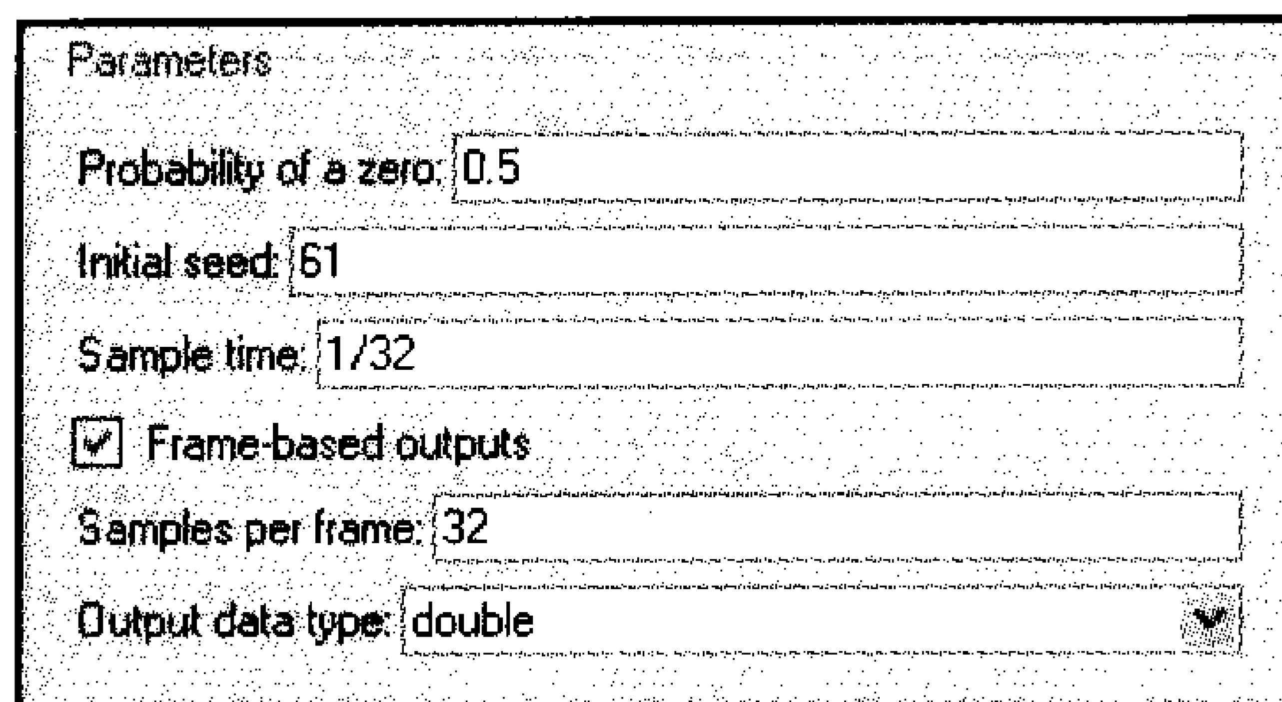


图 6-42　Bernoulli Binary Generator 模块参数设置

1．调试断点设置

在想要调试的模块前设置断点，系统执行到断点处暂停，并显示相应的调试信息。这里在 General CRC Generator 模块前设置断点。选中该模块，单击调试器工具栏中的【在选中模块前设置断点】按钮。此时，断点显示框中将显示断点设置情况，如图 6-43 所示。断点显示框中显示所有已设置了断点的模块。每个模块都有两个选项：第一个表示是否在该模块前设置断点；第二个表示是否显示选中模块的 I/O 信息。

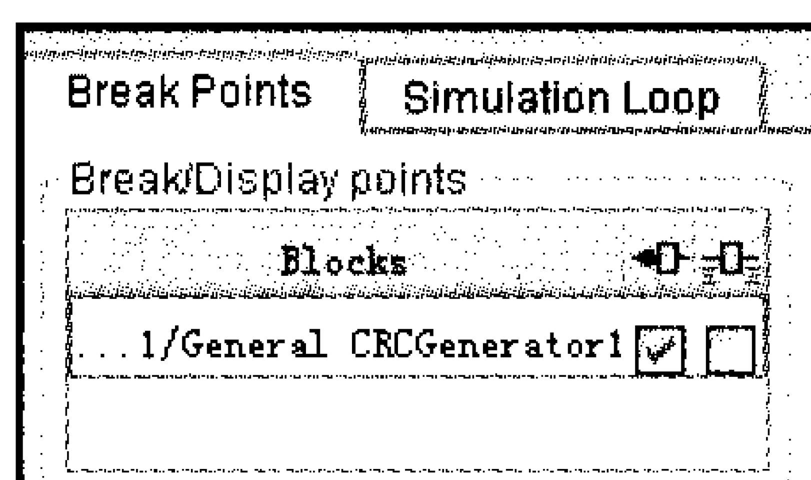


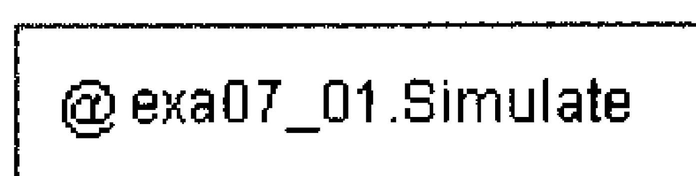
图 6-43　在 General CRC Generator 模块前设置断点

2.【开始/继续】按钮调试

当第一次单击【开始/继续】按钮后，Simulink 模型窗口左上方出现红色方框，框中的文字为“@exa07_01.Simulate”，如图 6-44 所示。这时表示模型调试处于准备就绪状态。同时在 Simulation Loop 窗口中显示黄色高亮的 exa07_01.Simulate 标题信息；在 Outputs 窗口中自动增加如下内容：

```
%----------------------------------------------------------------%
[TM = 0                                ] exa07_01.Simulate
(sldebug @0):
```

其中，TM = 0 表示当前仿真时刻为 0；sldebug @0 表示即将被执行的模块为当前系统的第一个模块 Bernoulli Binaiy Generator（序号为 0）。



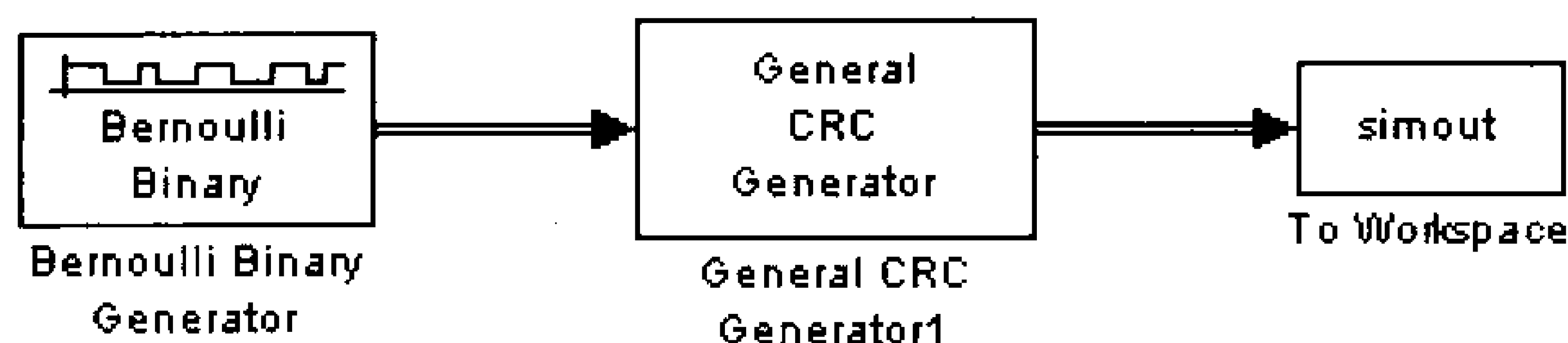


图 6-44　第一次单击【开始/继续】按钮后的 Simulink 模型窗口

再次单击【开始/继续】按钮，系统将会执行到下一断点处，即执行到已指定的 CRC 模块前的断点处。此时红色实方框变成粉红色的虚线方框，并指向 General CRC Generator 模块，如图 6-45 所示。同时，在 Simulation Loop 窗口中显示黄色高亮 exa07_01.Simulate、exa07_01.Start、RootSystem.Start 和 S_Function.Start 标题信息，如图 6-46 所示；在 Outputs 窗口中自动增加如下内容：

```
At break point:0 before 0:4 S-Function block methods 'exa07_01/General CRC Generator1'
%----------------------------------------------------------------%
[TM = 0                    ] 0:4 S-Function.Start 'exa07_01/General CRC Generator1'
(sldebug @6):
```

上述信息表示模型调试暂停在 General CRC Generator 模块的 S-Function 前，并表明模型调试已经执行了 6 个子模块（函数）了。

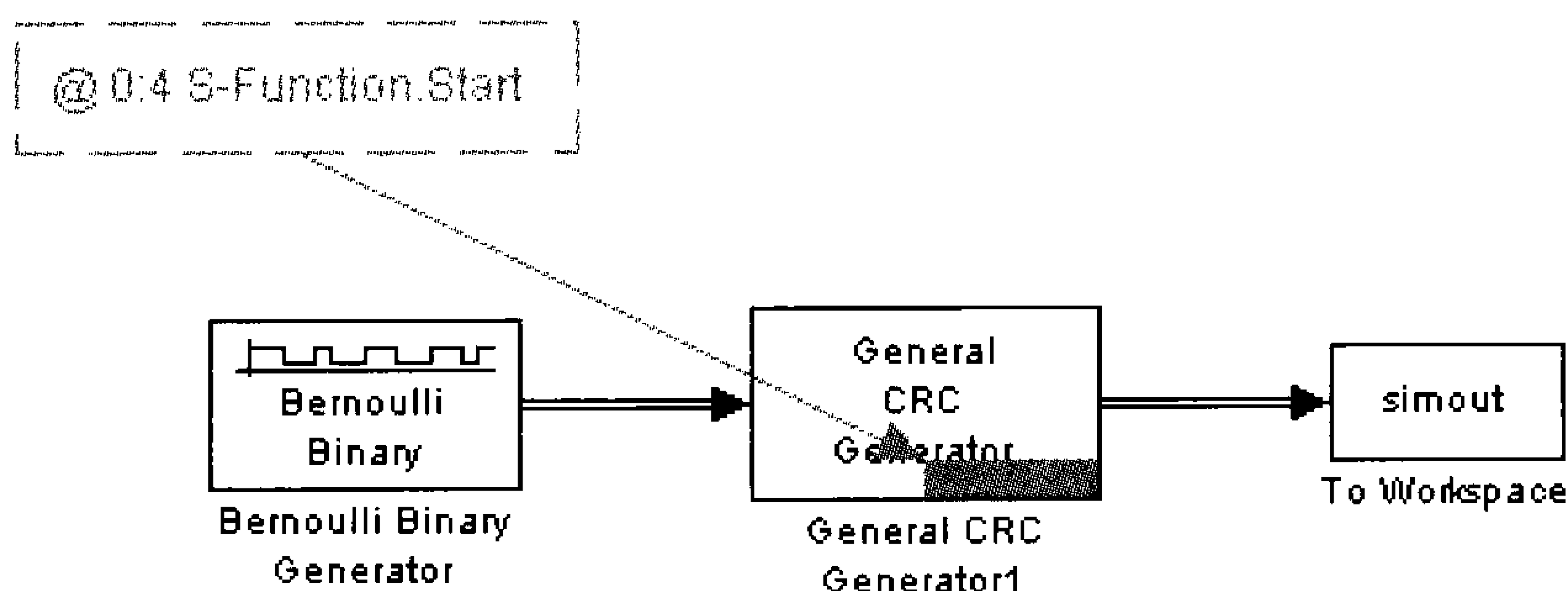


图 6-45　第二次单击【开始/继续】按钮后的 Simulink 模型窗口

图 6-46　执行到 General CRC Generator 模块前断点时的 Simulation Loop 窗口信息

可以单击【开始/继续】按钮继续调试，在上述窗口中将自动显示相关的信息，后面的操作请读者一步步亲身体验。

3．逐个模块调试

有时在系统调试过程中，不能确定是哪个模块存在问题，这时就需要逐个模块进行调试，观察每个模块的输入/输出。系统模型进入调试模式后，使用调试器工具栏中的【执行

下一模块】按钮，便可以逐个调试系统的模块。

4．逐时间步长调试

Simulink 允许用户使用逐时间步长的方式对系统进行调试。逐个模块调试可使用户详细观测系统中任何模块的输入/输出，但是这种调试方法非常耗时，不适合调试大型复杂系统。逐时间步长调试的特点是调试的时间间隔与系统仿真步长一致，并且在系统断点处停止调试。可以通过调试器工具栏中的“进入当前方法”、“跨过当前方法”、“暂时离开当前方法”和“在下一个时间步长返回第一个方法”等 4 个按钮实施模型的逐时间步长调试方法。

5．调试执行顺序与调试状态

在系统调试过程中，Simulink 调试器按照一定的顺序对系统模块进行调试。使用调试器的 Sorted List 窗口可以显示系统模块的调试顺序。本例中的模块调试顺序如下：

```
---- Sorted list for 'exa07_01' [6 nonvirtual blocks, directFeed=0]
  0:0    'exa07_01/Bernoulli Binary Generator/Random Source' (S-Function: sdsprandsrc2)
  0:1    'exa07_01/Bernoulli Binary Generator/DSP Constant/Constant' (Constant)
  0:2    'exa07_01/Bernoulli Binary Generator/Relational Operator' (RelationalOperator)
  0:3    'exa07_01/Bernoulli Binary Generator/Reshape' (Reshape)
  0:4    'exa07_01/General CRC Generator1' (S-Function: scomcrcgen)
  0:5    'exa07_01/To Workspace' (ToWorkspace)
```

从上面的信息可以看出，模块索引的形式是 *s*:*b*，其中 *s* 表示当前模块在被调试模型中的系统标号，而 *b* 则表示当前模块在该系统中的模块序号。

调试器的状态可以通过 Status 窗口进行观察，本例调试器状态如下：

```
%----------------------------------------------------------------%
Current simulation time                        : 0 (MajorTimeStep)
Default command to execute on return/enter : ""
Break at zero crossing events                  : disabled
Break on solver error                          : disabled
Break on failed integration step               : disabled
Time break point                               : disabled
Break on non-finite (NaN,Inf) values           : disabled
Break on solver reset request                  : disabled
Display level for disp, trace, probe           : 1 (i/o, states)
Solver trace level                             : 0
Algebraic loop tracing level                   : 0
Animation Mode                                 : off
Window reuse                                   : not supported
Execution Mode                                 : Normal
Display level for etrace                       : 0 (disabled)
Break points                                   : none installed
Display points                                 : none installed
Trace points                                   : none installed
```

6.2.3 使用命令行方式调试模型

对于 Simulink 高级用户而言，使用命令行调试模型会更加灵活和方便。下面将介绍使用命令行调试模型的方法。

1．启动/结束调试模式

在命令行中使用 sim 命令或 sldebug 命令来启动模型的调试模式。对于例 6.1 中建立的模型，可以使用如下命令：

```
>> sim('exa07_01',[0,10],simset('debug','on'))
```

或者

```
>> sldebug 'exa07_01'
```

在保证 exa07_01.mdl 目录为 MATLAB 当前工作目录的情况下，此时 MATLAB 会自动打开 Simulink 模型，并使处于调试模式的准备就绪状态，模型窗口如图 6-44 所示。同时，在 MATLAB 命令窗口中会显示如下信息：

```
%----------------------------------------------------------------%
[TM = 0                                ] exa07_01.Simulate
(sldebug @0): >>
```

这些信息与图形界面调试信息是一致的。这时，用户可以通过在提示符后输入模型调试命令，如设置断点、单步运行或显示仿真信息等命令。

在调试状态下无法直接关闭模型。如果需要关闭模型，首先要结束调试模式。在图形界面中通过单击【停止】按钮来实现；在命令行方式下，可以输入 stop 命令来结束调试。

2．设置/清除断点

由前面分析可知，Simulink 调试器可以设置两种类型的断点：无条件断点和有条件断点。如表 6-13 所示列出了可以设置断点的调试命令以及其功能。

表 6-13 断点设置命令及其功能

调试命令	功能	调试命令	功能
break<geb | s:b>	在选中或指定序号的模块前设置断点	rbreak	仿真需要重新设置解算器时
bafter<gcb | s:b>	在选中或指定序号的模块后设置断点	nanbreak	在数据溢出或无穷大出现时成为断点
tbreak [t]	在第 t 个仿真时间步长设置断点	xbreak	决定仿真步长的状态出现时成为断点
ebreak	在解算器出现错误处设置断点	zcbreak	过零事件发生之前成为断点

1）无条件断点

表 6-13 中左侧 3 个命令是用来设置无条件中断的断点的。无条件断点是一种固定断点，无论仿真如何进行，只要执行到断点处仿真都会中断。在模型处理调试状态下，可以选中想要设置成断点的模块，然后使用如下命令来设置断点：

```
>>break gcb
```

MATLAB 命令窗口中将显示如下信息，表明断点设置成功：

```
Installed break point:0 before 0:4 S-Function block methods 'exa07_01/General CRC Generator1'
```

或者使用模块序号来设置断点，使用如下命令，在索引号为 0:5 的模块前设置断点：

```
>>break 0:5
```

bafter 的使用方法与 break 相同，它是在指定或选中的模块后设置断点。

beark[t]表示在指定的第 t 个时间步长后设置断点（这里的 t 是真实的仿真时间而不是时间步长的序号，它可以是小数），例如，在 MATLAB 命令窗口输入：

```
>> tbreak 4
```

MATLAB 命令窗口中显示的信息为：

```
Time break point                                    : enabled (t>=4)
```

运行仿真，输入：

```
>> continue
```

这时，当遇到断点时，则调试处于暂停状态，并显示相关信息。如遇到 tbreak 4 设置的断点之后，MATLAB 命令窗口将显示如下信息：

```
%----------------------------------------------------------------%
[TM = 4                          ] exa07_01.Outputs.Major
(sldebug @19): >>
```

清除断点的命令为 clear，用来清除选中或指定序号的模块的断点。使用方法仍和 break 相同。例如：

```
>> clear gcb     %清除选中模块的断点
>> clear 0:5     %清除序号为 0:5 的模块的断点
```

2）有条件断点

表 6-13 中左侧的最后一个命令和右侧的 4 个命令用来设置条件中断的断点。与无条件断点不同的是，有条件断点在仿真中的具体时机不确定，只有当条件满足时断点才会出现，而且出现的位置也不一定相同，它受仿真运行时具体情况的影响。

nanbreak 命令用来设置在仿真中出现无限大的值时发生中断。如果在仿真中计算出了无穷大或超出了仿真计算的限制，即出现了上溢或下溢，调试器将会中断仿真。

xbreak 命令用来设置变步长解算器的中断。若模型使用变步长解算器，而且解算器遇到一个限制其使用变步长的状态，调试器就中断仿真。

zcbreak 命令在出现过零现象时发生中断。当 Simulink 检测到一个非采样零点出现在模型里，或者是模型包括可能产生零点的模块时，调试器就暂停仿真。暂停之后，会在命令窗口显示中断在模型中的位置、时间和过零点的类型（上升还是下降）。

ebreak 和 rbreak 命令这里不再赘述。

3．使用单步执行命令运行仿真

单步执行包括从一个模块执行到下一个模块、从一个时间步长执行到下一个时间步长，或者从一个断点执行到下一个断点来进行仿真调试。单步执行的调试命令如表 6-14 所示。

表 6-14　单步执行的调试命令

调 试 命 令	执行调试的方式	调 试 命 令	执行调试的方式
step	执行至下一个模块	Continue	执行至下一个断点处
Next	执行至下一个时间步长	Run	执行仿真至终点，忽略中间的断点

用 step 命令相当于图形调试器中的【执行至下一模块】按钮完成的功能，用来进行逐模块调试。

Next 命令相当于【执行至下一时间步】按钮完成的功能。这个命令允许用户只通过一个命令直接跳到下一个仿真时间步，并暂停在下一个仿真时间步长要执行的第一个模块的开始处。

Continue 命令完成执行到下一个断点的功能，与图形调试器的“开始/继续”按钮的功能相同。

Run 命令能够让系统忽略余下的所有断点，直接运行仿真至结束。如果用户对整个仿

真过程余下的部分没有兴趣，就可以使用该命令。

4. 显示模块的输入/输出

调试的一个重要手段就是观察模块的输入/输出数据，通过数据的观察来判断该模块是否工作正常。MATLAB 提供了 3 个命令来显示指定模块的输入/输出信息，它们分别是：disp、probe 和 trace，其主要区别在于显示输入/输出数据的时机不同。

1）disp 命令

disp 命令的作用是用来跟踪指定的一个和多个模块在用户单步执行仿真时的输入/输出数据。被 disp 命令指定的模块，其数据信息会在 MATLAB 每一次单步执行暂停时显示。

disp 命令的使用方法仍然和 break 命令一样，可以使用 disp gcb 和 disp s:b 两种形式对模块进行设置。用户可以使用 undisp 命令将一个模块从显示列表中去掉，它同样支持 gcb 和 s:b 两种指定方式。例如：

```
>> disp 0:5
```

执行后出现如下提示，表明设置成功：

```
Installed data display of 0:5 ToWorkspace block 'exa07_01/To Workspace'.
```

2）probe 命令

probe 命令则是用于显示非当前执行模块的输入和输出数据。在每次模型暂停调试时，用户就可以使用 probe 命令。probe 的使用方法与 disp 相同。具体如下：

```
>> probe 0:5
```

执行后出现如下信息：

```
probe: Data of 0:5 ToWorkspace block 'exa07_01/To Workspace':
U1      =
（……省略数值）
(sldebug @0): >>
```

3）trace 命令

使用 trace 命令指定的模块，用户可以在不中断仿真的情况下实时地观察其输入/输出数据。trace 的使用方法与 disp 和 probe 两种方法相同。使用 untrace 可以取消由 trace 指定模块输入/输出数据信息的显示。

5. 显示模块索引

在命令行调试的过程中经常要用到模块的索引信息。用户可以通过 slist 命令来显示模块的索引。对于刚才讲述的例子，在 MATLAB 命令窗口中输入

```
>> slist
```

命令时，在命令窗口中将会出现如下的模块索引信息：

```
---- Sorted list for 'exa07_01' [6 nonvirtual blocks, directFeed=0]
  0:0    'exa07_01/Bernoulli Binary Generator/Random Source' (S-Function: sdsprandsrc2)
  0:1    'exa07_01/Bernoulli Binary Generator/DSP Constant/Constant' (Constant)
  0:2    'exa07_01/Bernoulli Binary Generator/Relational Operator' (RelationalOperator)
  0:3    'exa07_01/Bernoulli Binary Generator/Reshape' (Reshape)
  0:4    'exa07_01/General CRC Generator1' (S-Function: scomcrcgen)
  0:5    'exa07_01/To Workspace' (ToWorkspace)
```

这些信息与图形调试中显示的信息是一致的。

6．其他调试命令

这里我们给出其他命令行调试命令和使用格式，读者可以自行尝试使用这些命令。

1）ashow 命令

ashow 命令用于高亮显示模型中的代数环。ashow 的命令格式如下：

```
ashow<gcb | s:b | s#n   | clear>
```

其中，

gcb：表示显示当前模块所在的代数环。

s:b：表示显示指定序号的模块所在的代数环。

s#n：表示显示系统中的第 n 个代数环。

Clear：则表示取消当前显示的代数环高亮状态。

2）atrace 命令

atrace 命令用于显示模型中的代数环在每个时间步长内被求解的有关信息。atrace 的命令格式如下：

```
atrace level
```

其中，level=0，1，2，3or 4

atrace 0：不显示信息。

atrace 1：求解代数环所需的迭代次数以及估计解的误差。

atrace2：同 atrace 1。

atrace 3：在 atrace 2 所显示的信息加上求解代数环路的 Jacobi 矩阵。

atrace 4：在 atrace 3 所显示的信息加上代数环路变量在每次迭代的瞬时解。

3）bshow 命令

该命令用来打开包含指定模块的系统，并选中该模块。它的命令格式如下：

```
bshow s:b
```

4）emode 命令

在加速和正常调试模式之间切换。

5）ishow 命令

开启或取消仿真过程中整数信息的显示。

6）minor 命令

开启或取消最小步长调试模式。在最小步长调试模式中，使用 step 命令进行逐个模块调试。当执行完当前主时间步长的最后一个模块后，如果仍然有小时间步长没有执行完，则 step 命令会使调试进入到下一个小时间步长内继续进行调试。直到所有的小时间步长执行完毕，step 命令会使调试进入下一个主时间步长的第一个小时间步长内。

7）quit 命令

退出当前仿真。

8）systems 命令

列出模型中所有的系统。

9）zclist 命令

列出包含非采样零点的模块。

6.3 Simulink 子系统建立与封装技术

随着系统规模和复杂性的增加，Simulink 模型中模块的数量也在不断地增大，模型中信息的主要流向也难以辨认。为了方便系统管理与使用，我们通常需要将部分模块按实现功能等指标组合成一个新的模块，对系统进行模块化设计，使其具有较强的层次性。换句话说，在建模和仿真过程中，我们需要建立和封装新的子系统。

6.3.1 子系统建立

用户可以通过以下两种方法建立子系统。

- 先添加模块，再将模块组合到子系统中。
- 先构建空白子系统，再把该子块所包含的模块添加进去。

1．组合已有的模块建立子系统

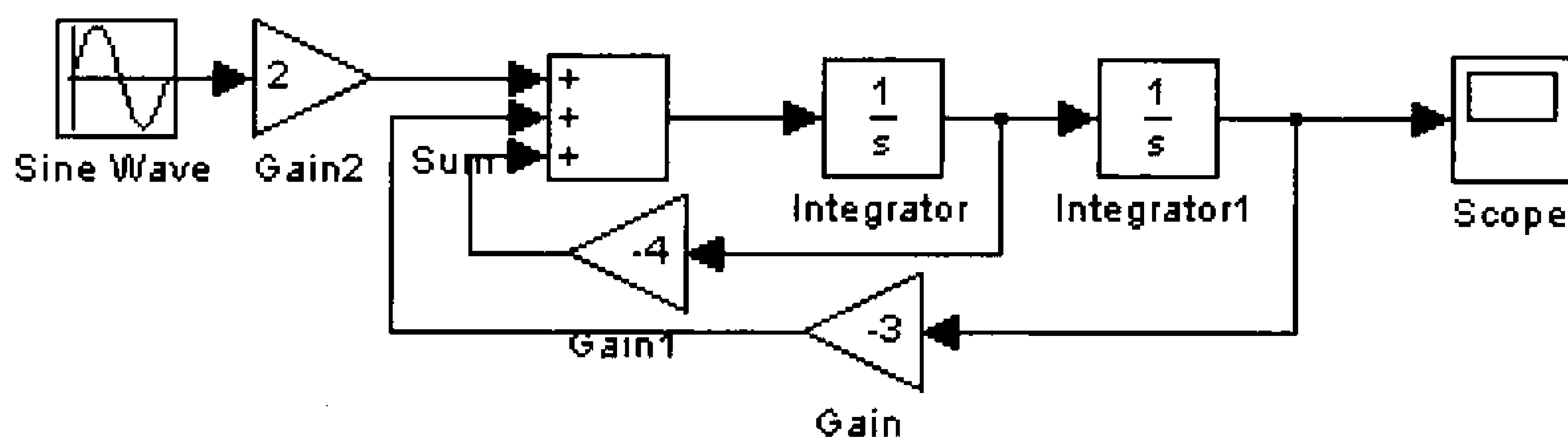
如果用户创建完了一些模块，又想把这些模块组合成子系统，其操作步骤如下所示。

（1）用方框同时选中待组合的模块或者按住【Shift】键逐个选中。

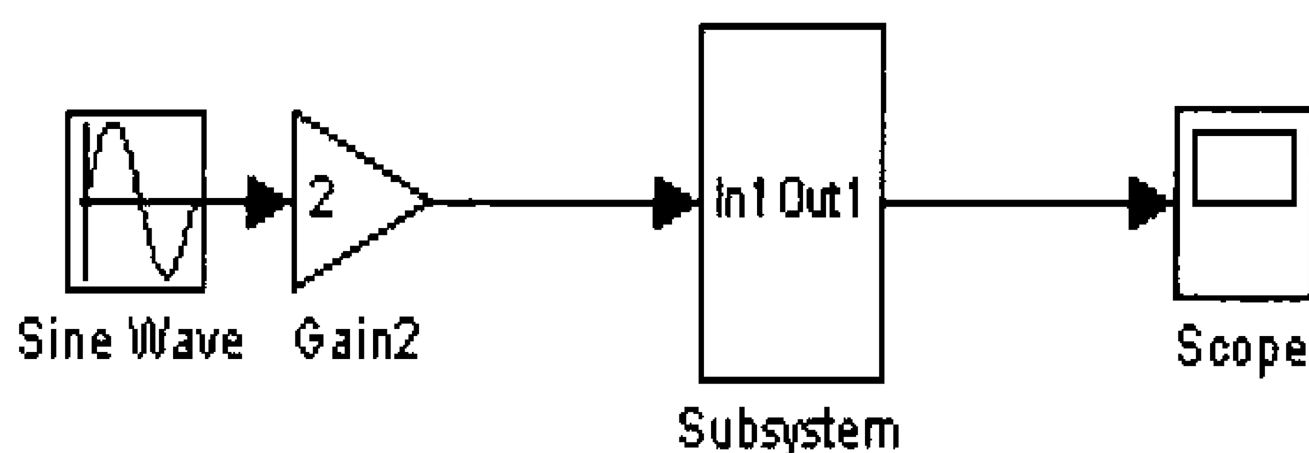
（2）选择【Edit】菜单或单击鼠标右键，在弹出的快捷菜单中执行【Create Subsystem】命令，子系统就创建成功了。

例 6.2 组合已有的模块建立子系统实例。

在如图 6-47（a）所示中，选中 Sum、Gain、Gain1、Integrator 和 Integrator1 等 4 个模块及其连线，执行【Edit】→【Create Subsystem】命令，即可创建子系统，得到如图 6-47（b）所示的模型。



（a）由模块组成的模型系统



（b）模块组合成子系统后的构成模型系统

图 6-47 组合已有的模块建立子系统

2．用空白 Subsystem 模块建立子系统

如果需要在模型中新建立一个子系统，模型本身不包含组成子系统的模块，可以按下列步骤进行。

（1）在“Simulink Library Brower”中打开 Simulink 库，从其中的 Ports&Subsystems 库中选取合适的 Subsystem，并拖至模型窗口中。

（2）双击 Subsystem 模块，打开 Subsystem 窗口。

（3）把要组合的模块拖曳到 Subsystem 窗口中，然后在该窗口中加入 Inport 模块表示从子系统外部到内部的输入，加入 Outport 模块表示从子系统内部到外部的输出，并把这些模块按顺序连接起来，子系统就建立成功了。

例 6.3 用空白 Subsystem 模块建立子系统实例。

使用空白 Subsystem 模块，添加相应的模块后，建立如图 6-48 所示的子系统，其功能相当于图 6-47（b）中的 Subsystem 模块。

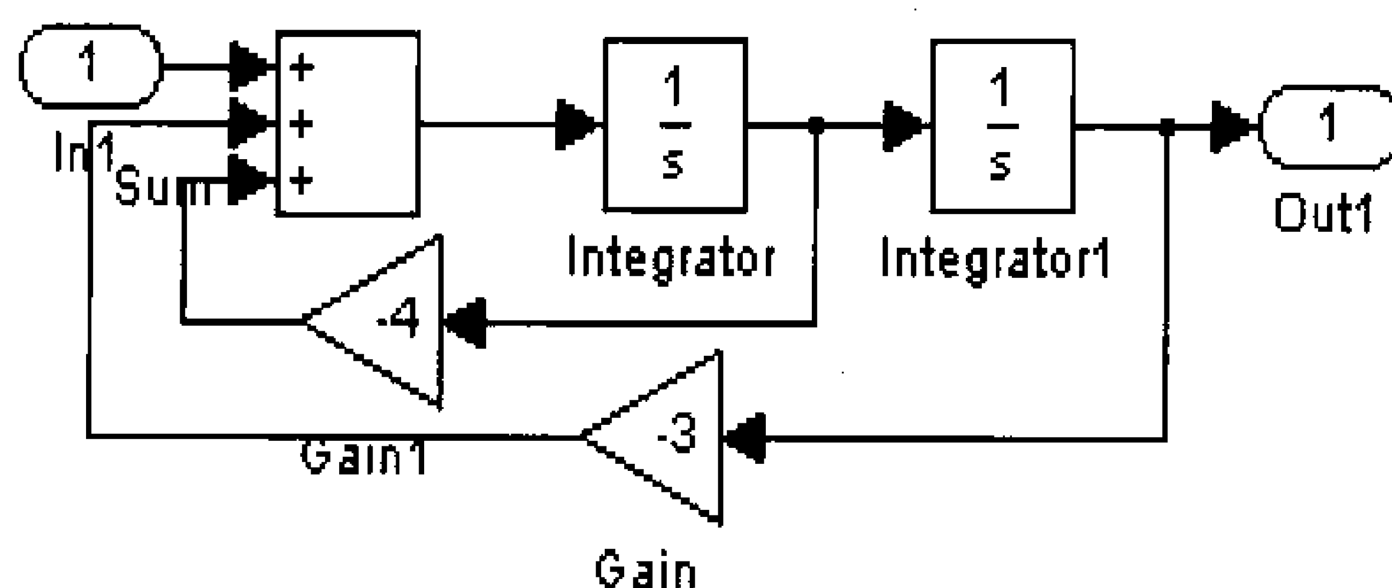


图 6-48 子系统中的模块

6.3.2 条件执行子系统建立

子系统的最基本目的就是将一组相关的模块包含到一个模块中，用以简化系统，使得系统的分析更加容易。例如在一个控制系统中，受控系统可视为一个子系统，控制器也可以作为一个子系统。这些子系统就如其他一般模块一样，都是具有特定输入输出的模块。给定输入信号，就可以产生相应的输出信号。但是对于某些特殊的情况，并不是对所有的输入信号都要产生输出信号，只有在某些特定的条件下才会产生输出信号，这时就需要输入一个控制信号。控制信号由子系统模块的特定端口输入，这样的子系统称为条件执行子系统。在条件子系统中，输出信号取决于输入信号和控制信号。

根据不同的控制信号，可将条件执行子系统分为以下两大类。

- 触发子系统：当控制信号的触发沿（上升沿、下降沿或双边沿等）有效时，执行子系统。
- 使能子系统：当控制信号为正时，执行子系统。

1．**触发子系统**（Trigger Subsystem）

触发子系统是指当触发事件发生时开始执行的子系统。它在外观上有一个“触发”控制信号输入口，仅当触发输入信号所定义的某个事件恰巧发生时，该模块才开始接受输入端的信号。子系统一旦被触发，其输出端口的值就保持不变，直到下次再触发才可能改变。

当触发信号是向量时，只要向量中有一个分量信号发生“触发事件”，子系统就会被触发。

“触发事件”由子系统内触发模块对话框定义，有以下 4 种触发事件形式可以选择。

- rising 上升沿触发：触发信号以增长的方式穿越 0 时，子系统开始接收输入值。
- falling 下降沿触发：触发信号以减小的方式穿越 0 时，子系统开始接收输入值。
- either 任意沿触发：每当触发信号穿越 0 时，子系统开始接收输入值。
- fuction-call 函数调用触发：这种触发方式必须和 S 函数配合使用。

能放置在触发系统中的模块包括对采样操作具有“继承”特性的逻辑模块和增益模块，采样时间被设置为–1 的离散模块。一般的连续时间模块不能置于触发子系统中。

例 6.4 触发子系统建模实例。

由触发子系统建立模型如图 6-49 所示，其 Trigger 触发模块有 3 种，即 rising 上升沿、falling 下降沿和 either 任意沿控制信号模块，从中可以发现这 3 种子系统的区别。

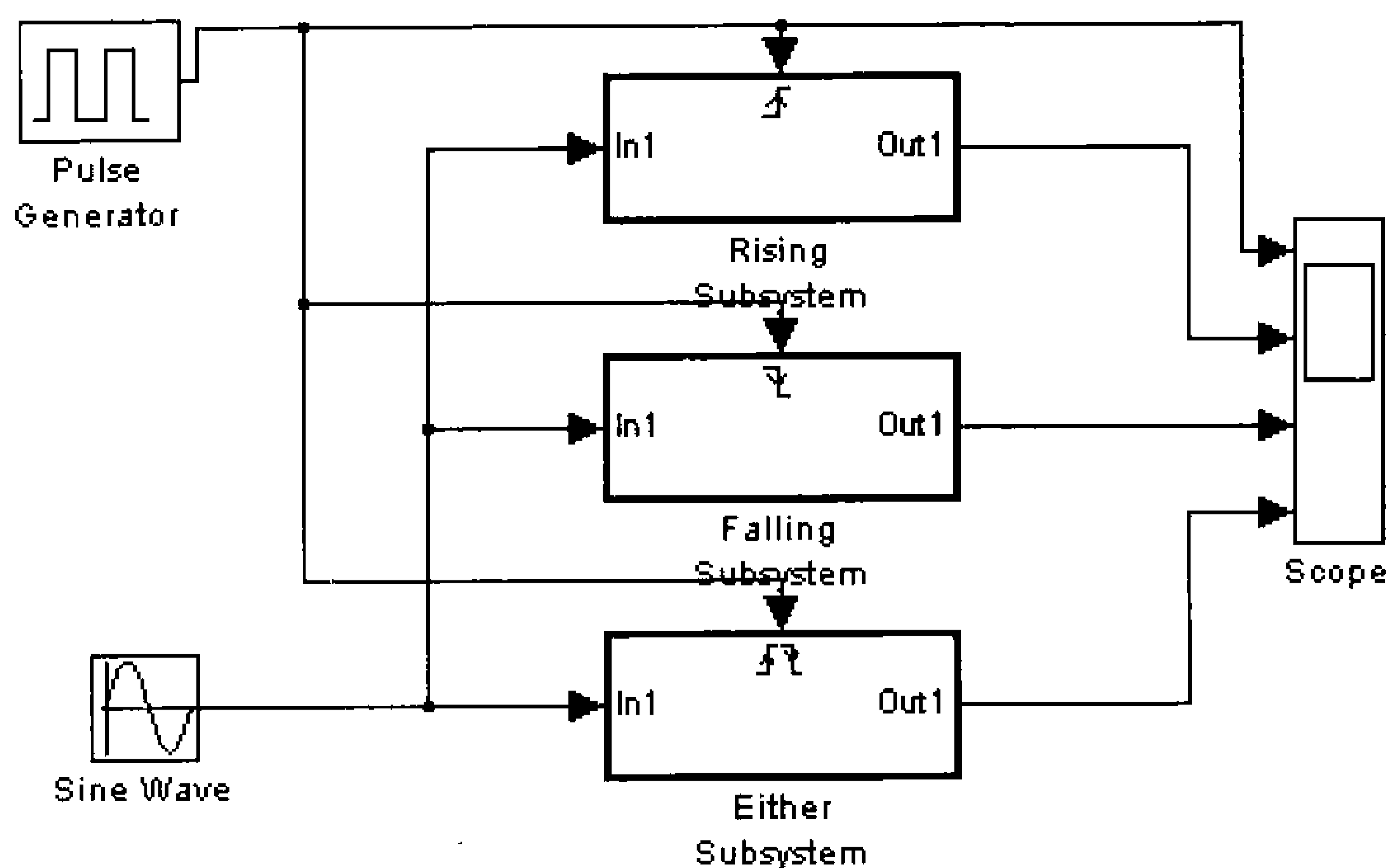


图 6-49 触发子系统建模

1）模块参数设置

①Pulse Generator 模块参数设置。双击该模块，在弹出的对话框中设置“Amplitude”为“1”，“Period”为“1”，“Pulse Width”为“50”，“Phase delay”为“0”，如图 6-50 所示。

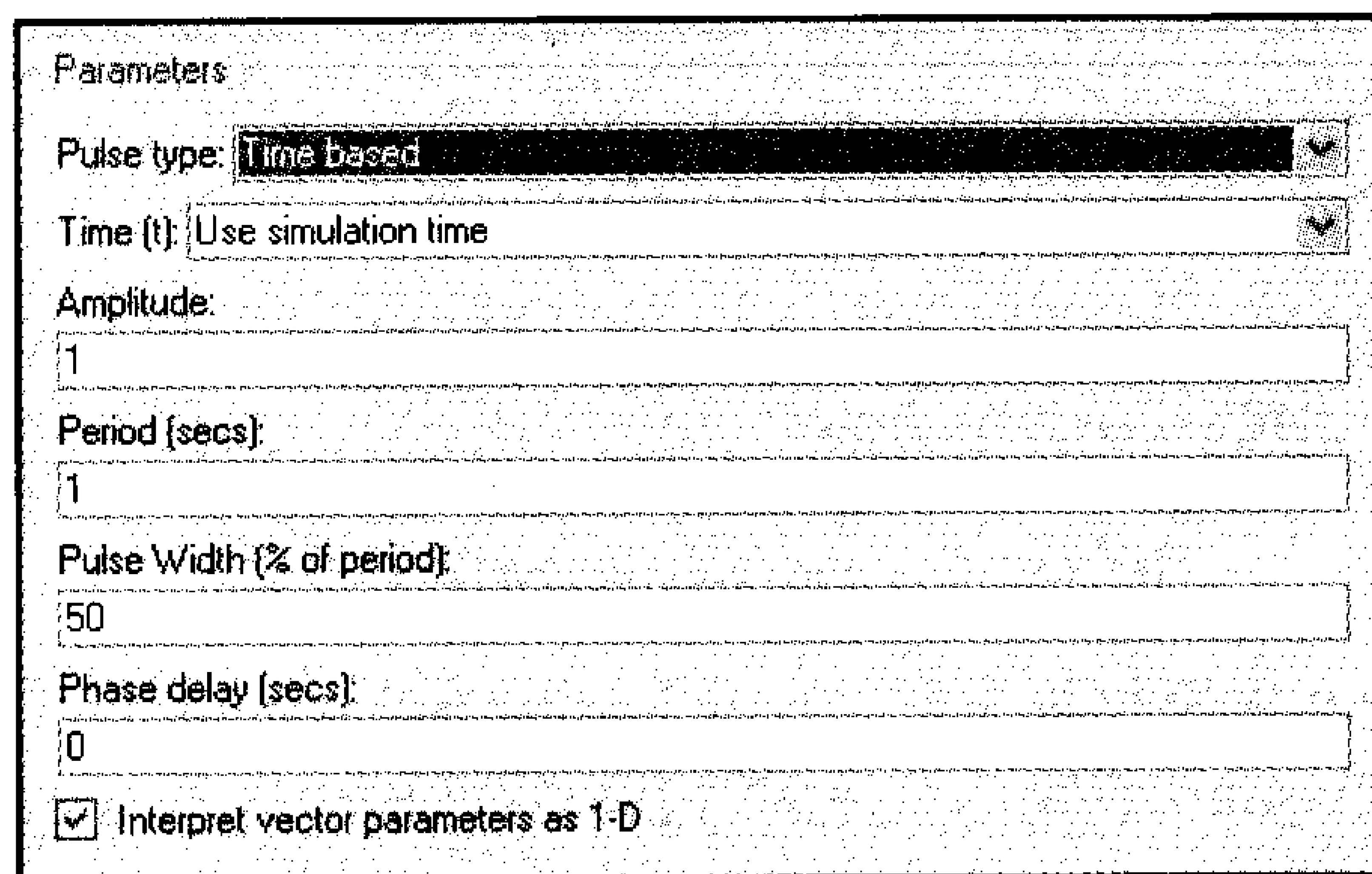


图 6-50 Pulse Generator 模块参数设置

②Sine Wave 模块参数设置。双击该模块，在弹出的对话框中设置“Amplitude”为“1”，

"Bias"为"0","Frequency"为"1","Phase"为"0","Sample time"为"0",如图 6-51 所示。

Parameters
Sine type: Time based
Time (t): Use simulation time
Amplitude:
1
Bias:
0
Frequency (rad/sec):
1
Phase (rad):
0
Sample time:
0
☑ Interpret vector parameters as 1-D

图 6-51　Sine Wave 模块参数设置

③Rising Subsystem 模块参数设置。打开 Rising Subsystem 子系统模型,如图 6-52 所示。然后双击其中的 Trigger 模块,弹出如图 6-53 所示的参数设置对话框,在该对话框中设置 Trigger Type 栏中选项为 rising,单击【OK】按钮。

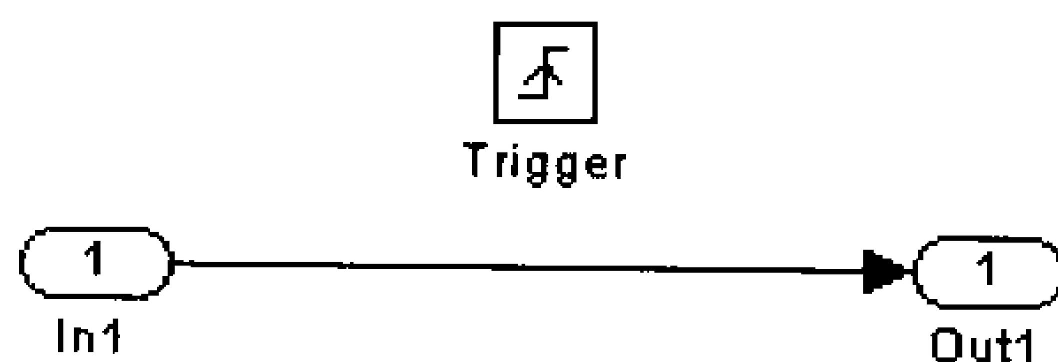


图 6-52　Rising Subsystem 触发子系统模型

Parameters
Trigger type: rising
States when enabling: held
☐ Show output port
Output data type: auto
Sample time type: triggered
Sample time:
1
☑ Enable zero crossing detection

图 6-53　Trigger 触发模块的参数设置

④Falling Subsystem 模块参数设置。打开 Falling Subsystem 子系统模型,如图 6-54 所示。然后双击其中的 Trigger 模块,在弹出的参数对话框中设置 Trigger Type 栏中选项为 falling,单击【OK】按钮。

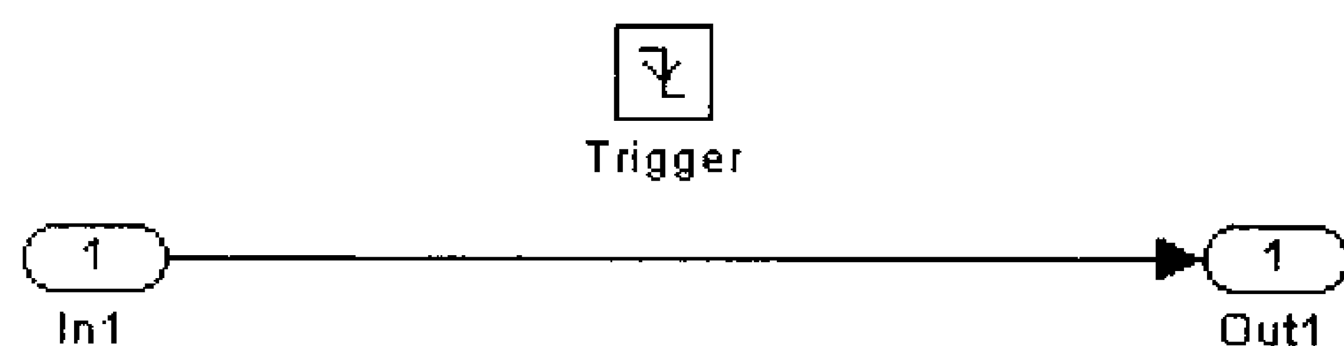


图 6-54　Falling Subsystem 触发子系统模型

⑤Either Subsystem 模块参数设置。打开 Either Subsystem 子系统模型，如图 6-55 所示。然后双击 Trigger 模块，在弹出的参数对话框中设置 Trigger Type 栏中选项为 either，单击【OK】按钮。

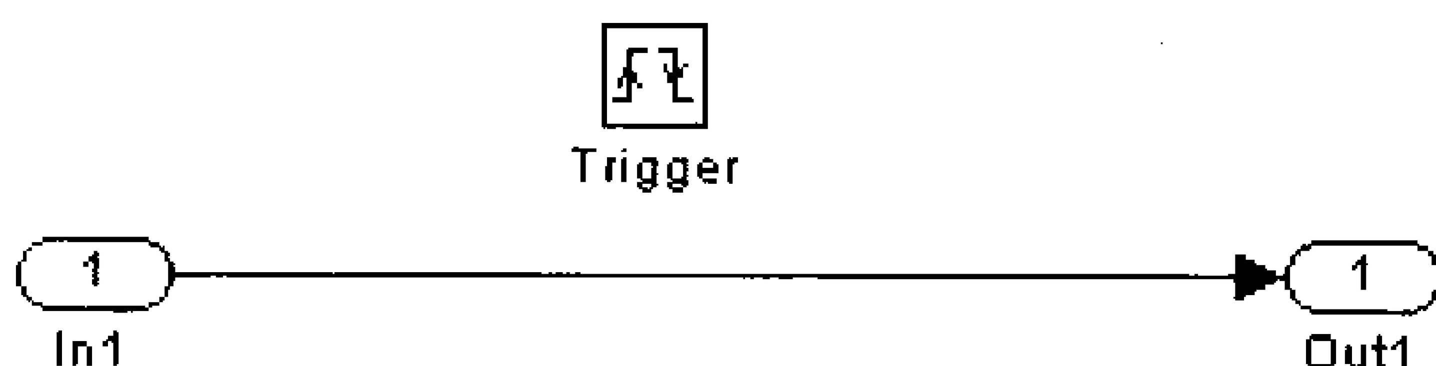


图 6-55　Either Subsystem 触发子系统模型

⑥scope 模块设置。双击此模块，在弹出窗口单击【参数设置】按钮，在弹出的对话框中设置 Number of axes 为 4。

2）仿真运行及结果分析

该触发子系统模型的运行结果如图 6-56 所示。在该图中，第一幅子图是触发信号；第二幅子图是上升沿触发子系统运行结果；第三幅子图是下降沿触发子系统运行结果；第四幅子图是双边沿触发子系统运行结果。

据图可知，在触发信号由 1 到 0 时，下降沿触发子系统和双边沿触发子系统被执行；当由 0 到 1 时，上升沿触发子系统和双边沿触发子系统被执行。

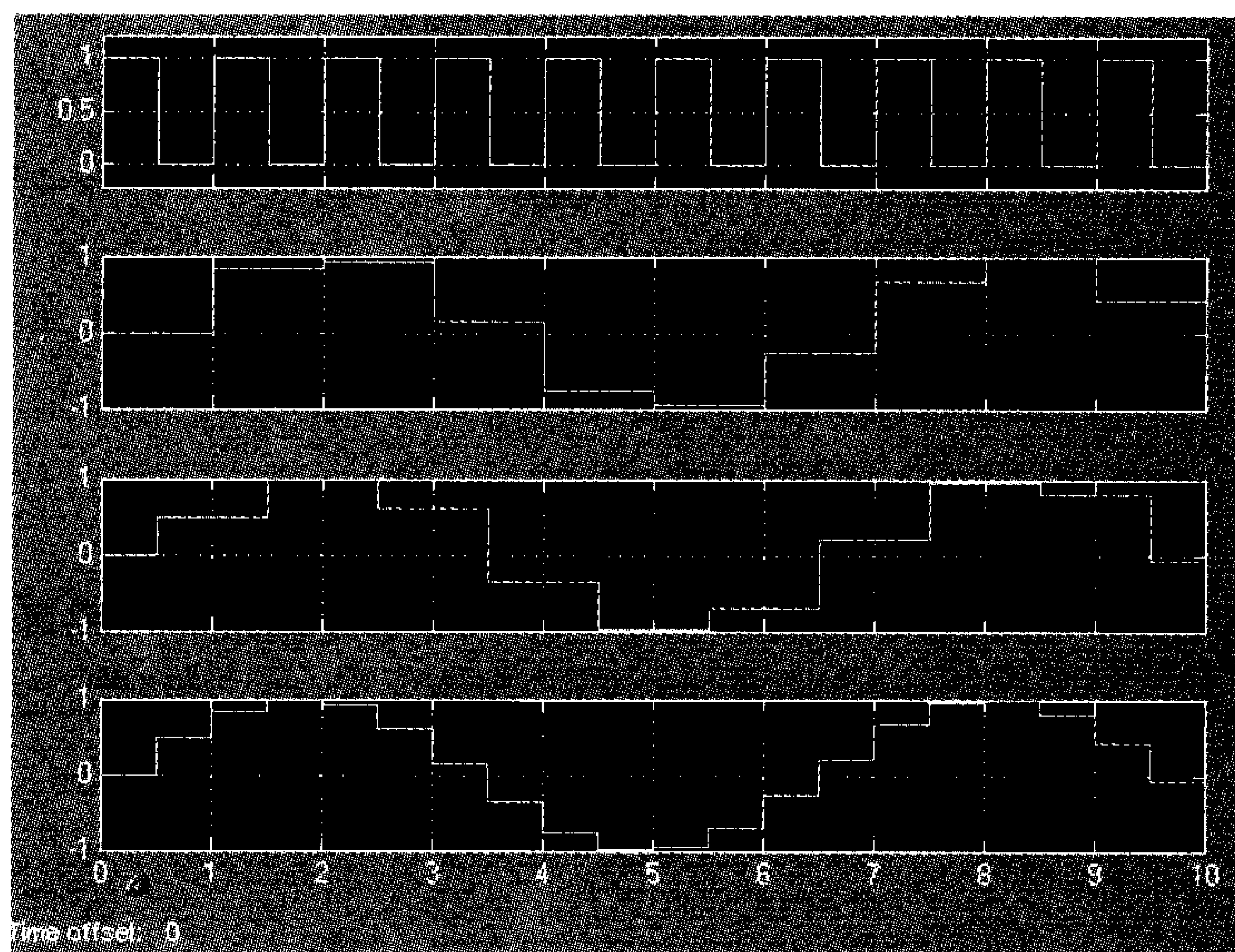


图 6-56　触发子系统模型仿真结果

2．使能子系统（Enable Subsystem）

使能子系统在控制信号从负数朝正向穿过零时开始执行，直到控制信号变为负数时停止。使能子系统的控制信号可以是标量也可以是向量。如果控制信号是标量，当该标量的值大于 0 时子系统执行；如果是向量，向量中的任意一个元素大于 0 时，子系统开始执行。

任何连续和离散模块都可以作为使能子系统。

例 6.5 使能子系统建模实例。

使能子系统模型如图 6-57 所示，在本例中使用了两个使能子系统，为了能够更加清晰地了解使能子系统的功能，对同一输入信号取截然相反的输入控制信号，对比子系统的输出。为了构造截然相反的输入控制信号，我们采用了 Gain 模块和 Constant 模块，当然，读者可自己采用不同的模块来构造。

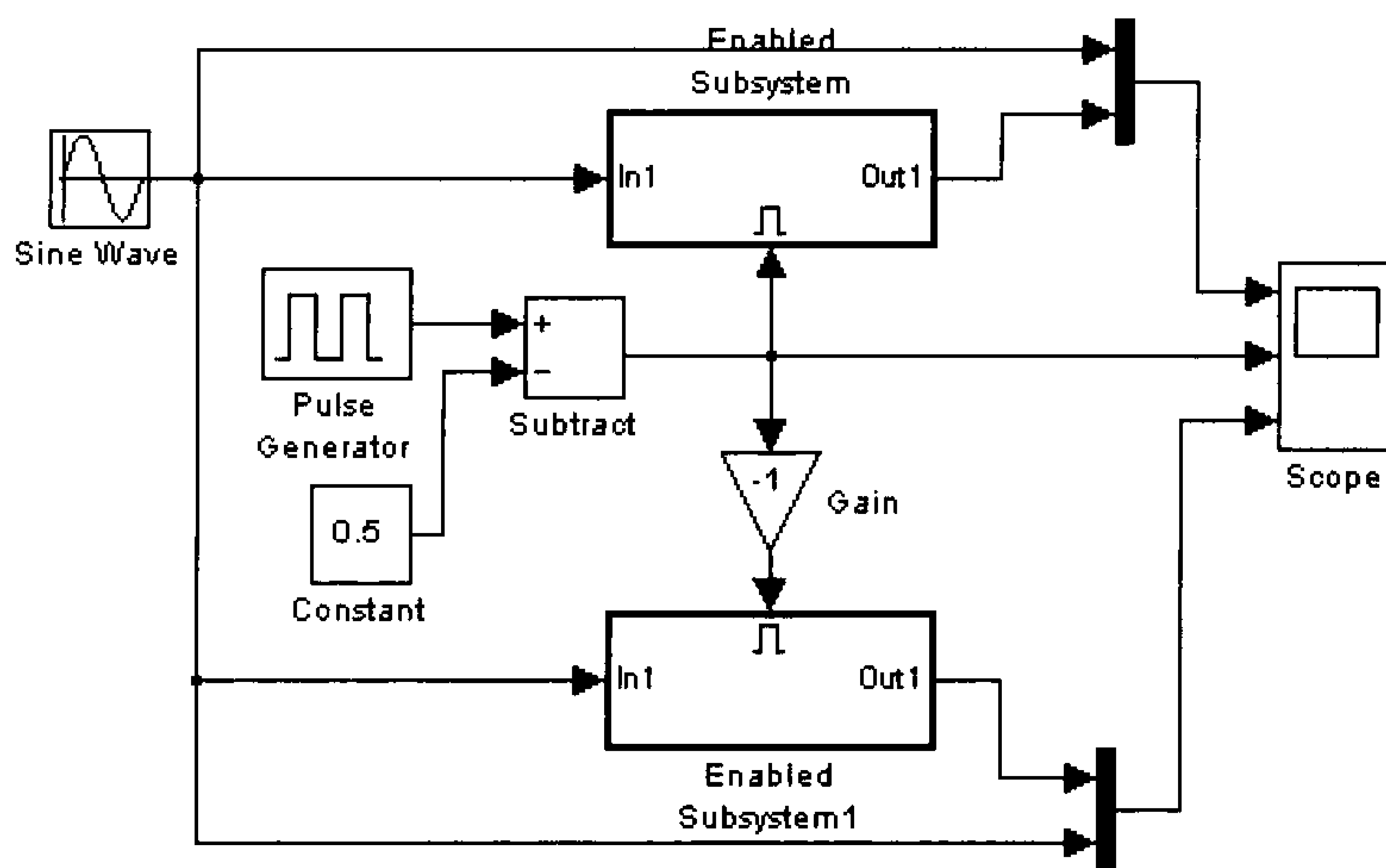


图 6-57 使能子系统的模型

1）模块参数设置

①Pulse Generator 模块参数设置。其中“Amplitude”为“1”，“Period”为“1”，“Pulse Width”为“50”，“Phase delay”为“0”。

②Sine Wave 模块参数设置。其中“Amplitude”为“1”，“Bias”为“0”，“Frequency”为“1”，“Phase”为“0”，“Sample time”为“0”。

③Constant 模块参数。其中“Constant value”为“0.5”。

④Gain 模块参数。其中“Gain”为“–1”。

⑤Enabled Subsystem 模块和 Enabled Subsystem1 模块采用同样的设置。打开子系统模型，如图 6-58 所示，然后双击 Enable 模块，进行参数设置，结果如图 6-59 所示。

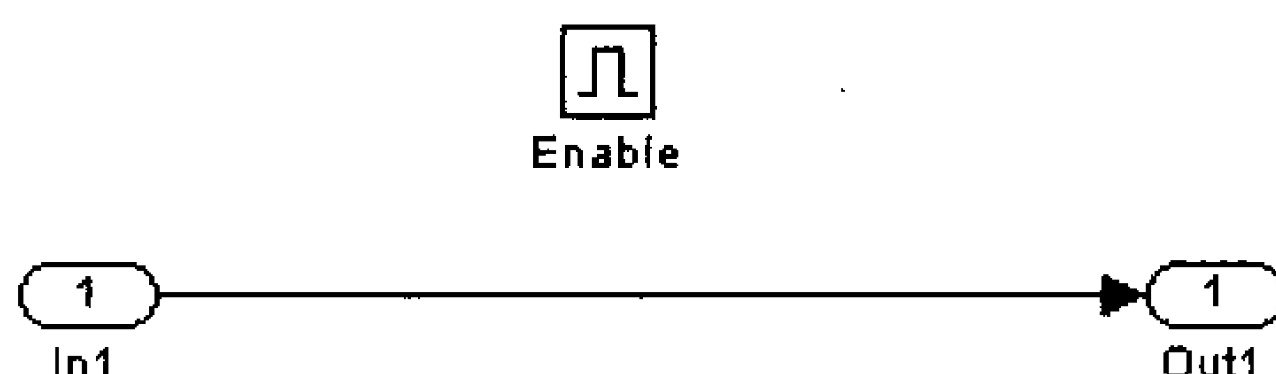


图 6-58 Enabled subsystem 子系统模块

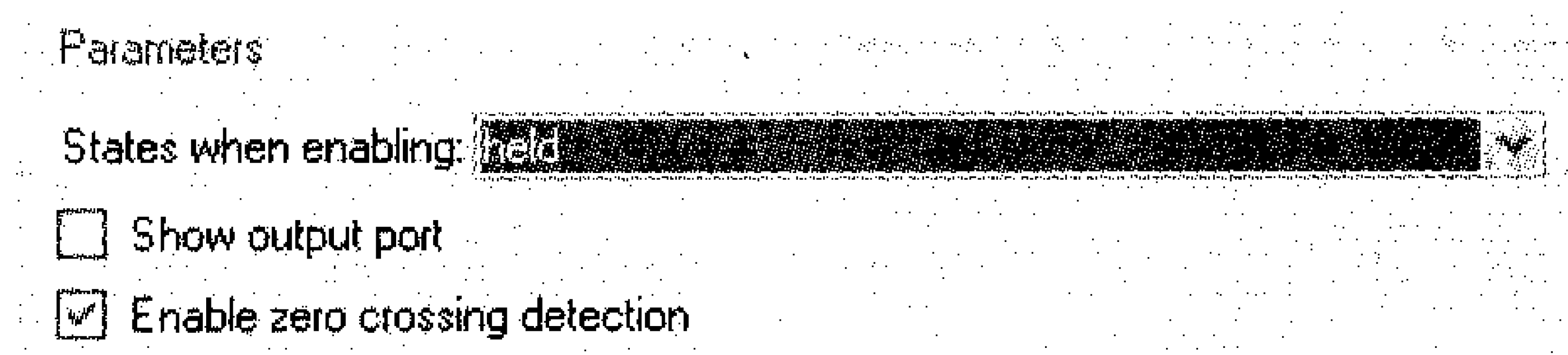


图 6-59 Enable 参数对话框



图 6-59 中“States when enabling”有“reset”和“held”两个选项：若选择“reset”，则“使能”时，将把所在子系统所有内部状态重置为指定的初值；若选择“held”，则“使能”时，将把所在子系统所有内部状态保持在“前次使能”的终值上。“Show output port”复选框若被选中，那么使能模块将出现一个输出口。从这个输出口输出控制信号，可以对控制信号进行监视和分析。勾选“Enable zero crossing detection ”复选框，则会启动探测零交叉的功能。

2）仿真运行及结果分析

模型系统参数采用默认值，仿真结果如图 6-60 所示。在该图中，第一幅子图为 Pulse Generator 模块和 Constant 求和信号直接输入使能模块的结果图和正弦信号图，第二幅子图为 Pulse Generator 模块信号，第三幅子图为 Pulse Generator 模块和 Constant 求和信号取反后输入使能模块的结果图和正弦信号图。

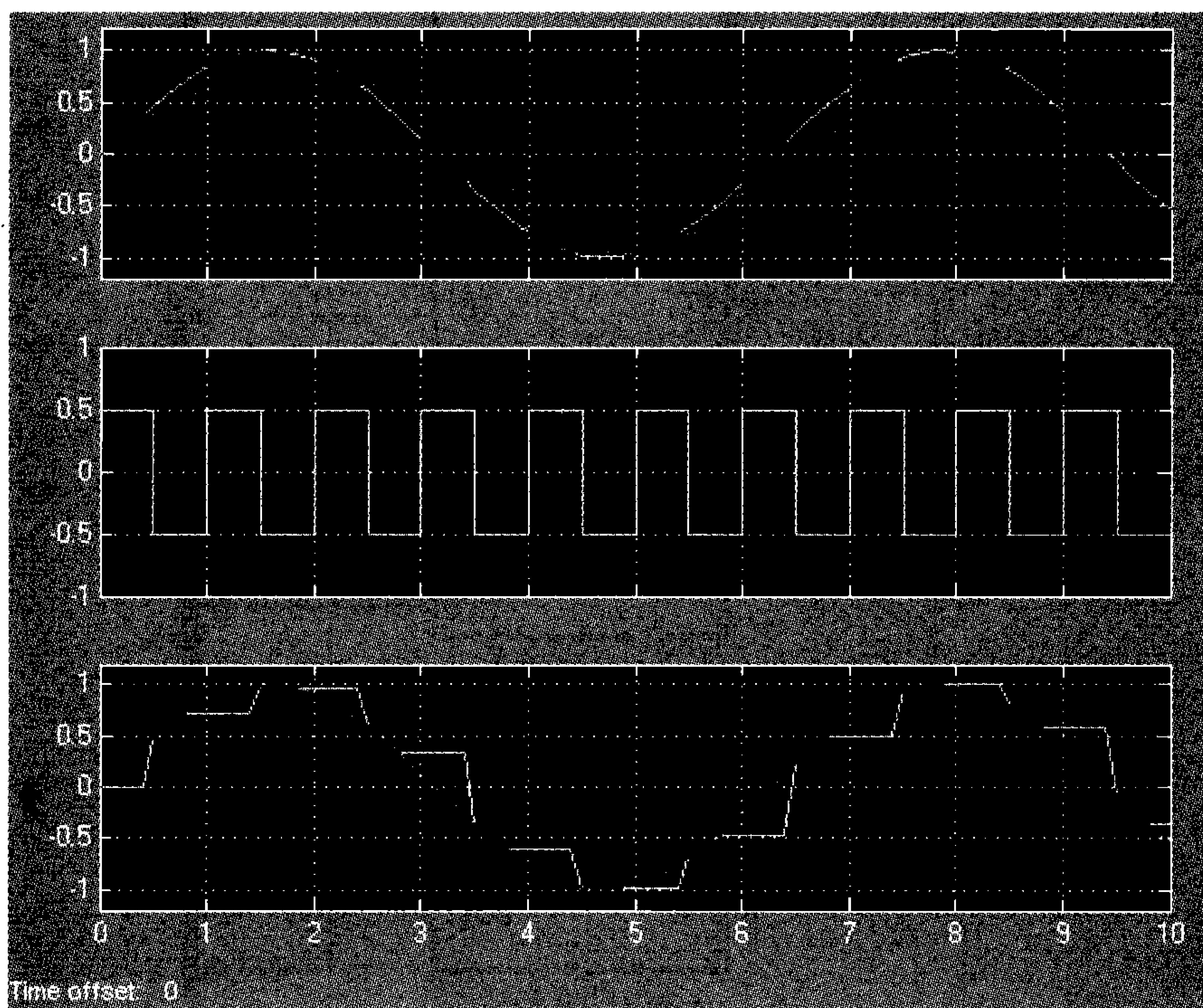


图 6-60 使能子系统模型仿真结果

从仿真结果可以看出，当Pulse Generator模块产生的信号为正时，第一个使能子系统直接输出正弦信号；而第二个使能子系统的信号则保持不变。当Pulse Generator模块产生的信号为负时，情况则正好相反，第二个使能子系统直接输出正弦信号，而第一个使能子系统的信号则保持不变。

3．触发使能子系统

使能加触发子系统就是使能子系统和触发子系统的组合。它把Trigger模块和Enable模块同时加入到子系统中就形成了使能加触发子系统。

使能加触发子系统含有触发信号和使能信号两个控制信号输入端。触发事件发生后，Simulink检查使能信号是否大于0，大于0即可开始执行。使能加触发子系统的参数可以分别进行设置，在Trigger模块中设置触发类型的参数，而在Enable类型中设置子系统再次开始执行时的状态值。此时，输出端口模块的参数设置和使能子系统相同。

例6.6 触发使能子系统建模实例。

为了可以比较触发使能子系统的功能，以及它们不同设置之间的差异，本例构建的触发子系统的模型如图6-61所示，它由4个触发使能子系统组成。为了获得截然相反的输入控制信号，我们采用Gain模块、Pulse Generator模块和Constant模块。

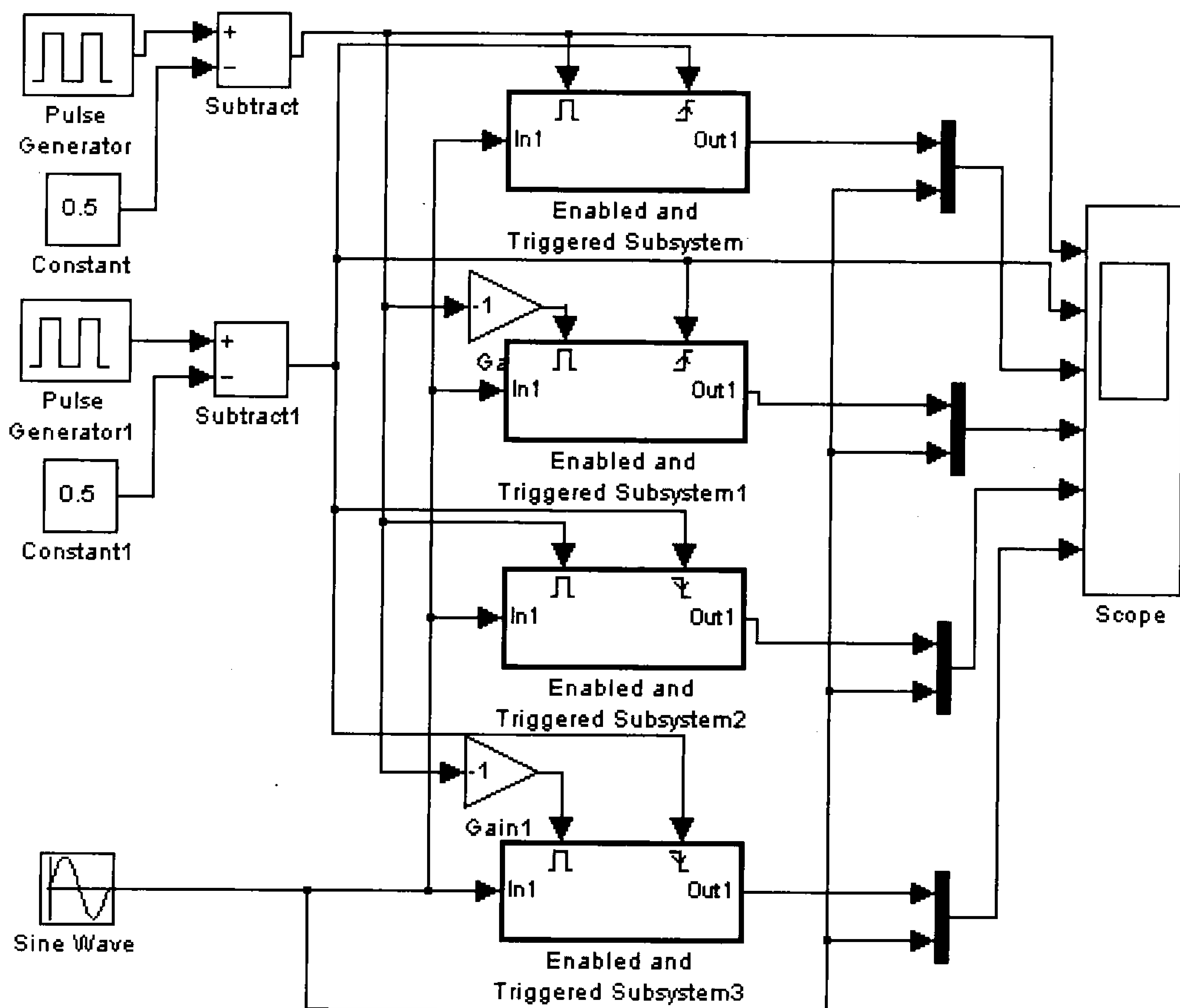


图6-61　触发使能子系统模型

1）模块参数设置

①Pulse Generator 模块参数设置。其中 “Amplitude” 为 “1”，“Period” 为 “1”，“Pulse Width” 为 “50”，“Phase delay” 为 “0”。

②Pulse Generator 模块参数设置。其中 “Amplitude” 为 “1”，“Period” 为 “0.5”，“Pulse Width” 为 “50”，“Phase delay” 为 “0”。

③Sine Wave 模块参数设置。其中 “Amplitude” 为 “1”，“Bias” 为 “0”，“Frequency” 为 “1”，“Phase” 为 “0”，“Sample time” 为 “0”。

④Constant 模块和 Constant 1 模块参数设置。其中 “Constant value” 为 “0.5”。

⑤Gain 模块和 Gain1 模块参数设置。其中 “Gain” 为 “–1”。

⑥Enabled and Triggered Subsystem 模块和 Enabled and Triggered Subsystem 1 模块参数设置。这两个模块采用同样的设置。打开这两个子系统模块，如图 6-62 所示，接着双击 Triggered 模块，在弹出对话框（如图 6-53 所示）的“Triggered type”下拉列表框中选择“rising”选项，然后双击 Enable 模块，在弹出对话框（如图 6-59 所示）的 “States when enabling” 下拉列表框中选择 “held” 选项。

⑦Enabled and Triggered Subsystem 2 模块和 Enabled and Triggered Subsystem 3 模块参数设置与步骤⑥基本相同，只在图 6-53 所示对话框中的 “Trigger type” 下拉列表框中选择 “falling” 选项。

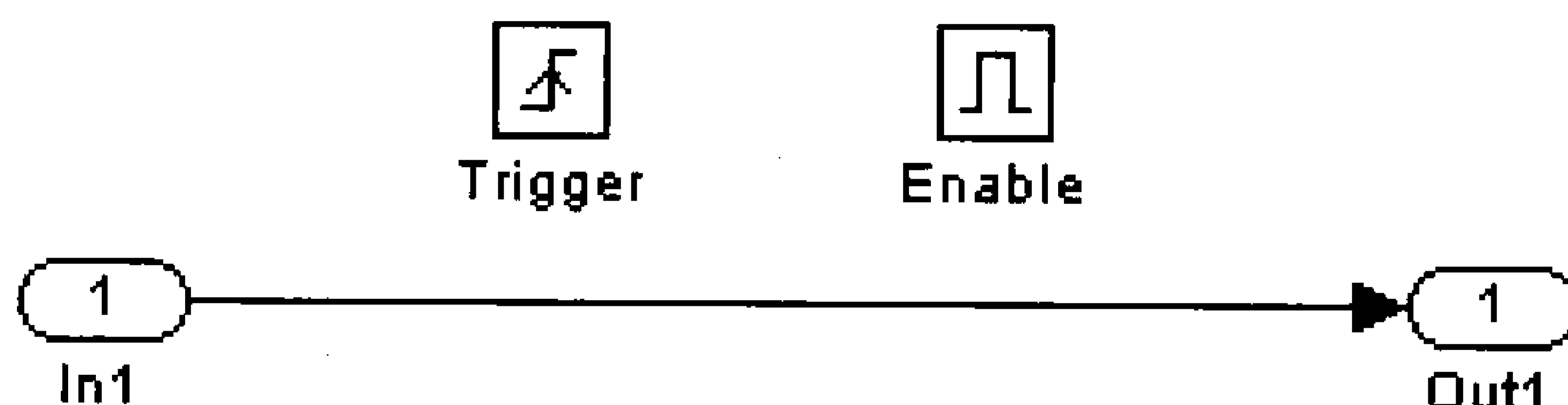


图 6-62　触发使能子系统模型

2）仿真运行与结果分析

在仿真过程中，模型系统参数采用默认值，仿真结果如图 6-63 所示。在该图中，第一幅子图是使能控制信号；第二幅子图是触发控制情号；第三幅子图和第四幅子图分别表示使能信号直接输入和取反输入时，并且触发控制信号为上升沿的情况下，使能触发模型系统的仿真结果；第五幅子图和第六幅子图分别表示使能信号直接输入和取反输入时，并且触发控制信号为下降沿的情况下，使能触发模型系统的仿真结果。

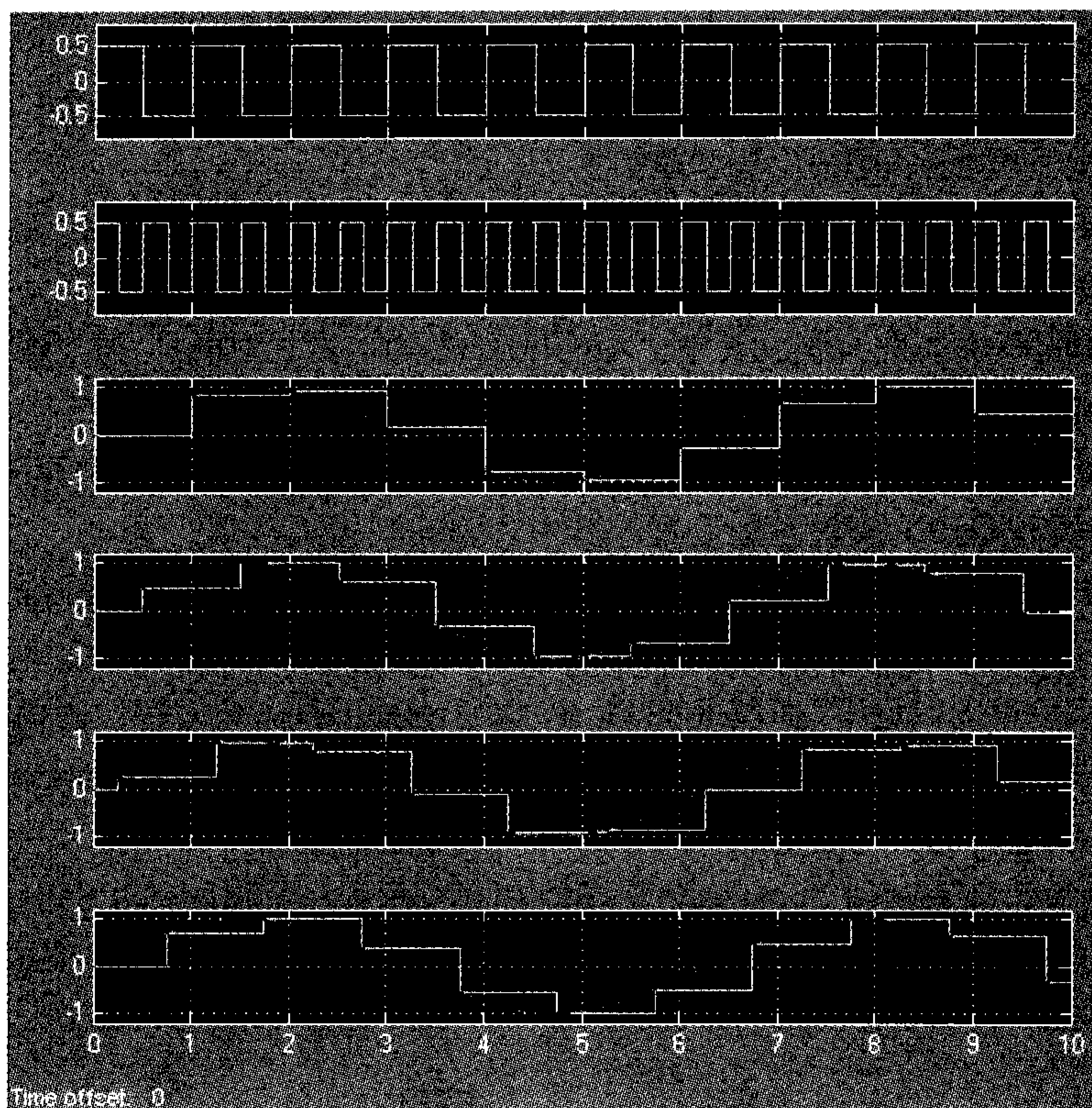


图 6-63　触发使能模型系统的仿真结果

据图 6-63，可以得出以下基本结论。

第三幅子图表明，在输入使能触发子系统模块的使能信号为正，并且触发信号为上升沿触发信号时，输出才会变化，其他情况都保持常值。

第四幅子图表明，在输入使能触发子系统模块的使能信号为负，并且触发信号为上升沿触发信号时，输出才会变化，其他情况都保持常值。

第五幅子图表明，在输入使能触发子系统模块的使能信号为正，并且触发信号为下降沿触发信号时，输出才会变化，其他情况都保持常值。

第六幅子图表明，在输入使能触发子系统模块的使能信号为负，并且触发信号为下降沿触发信号时，输出才会变化，其他情况都保持常值。

4. Switch Case 和 Switch Case Action Subsystem 子系统

Switch Case 子系统就是在对输入信号进行判断的基础上，然后选择相应的输出端口，使得相应的 Switch Case Action Subsystem 子系统被执行。下面通过一个实例说明这种子系统的使用方法。

例 6.7　Switch Case 和 Switch Case Action Subsystem 子系统建模实例。

建立的模型如图 6-64 所示，这是一个对 Switch Case 模块输入信号进行端口选择，然后执行相应的 Switch Case Action Subsystem 子系统，输入信号为 1，2，3，4，其中，4 个 Step 模块用来构造 Switch Case 模块的输入信号。

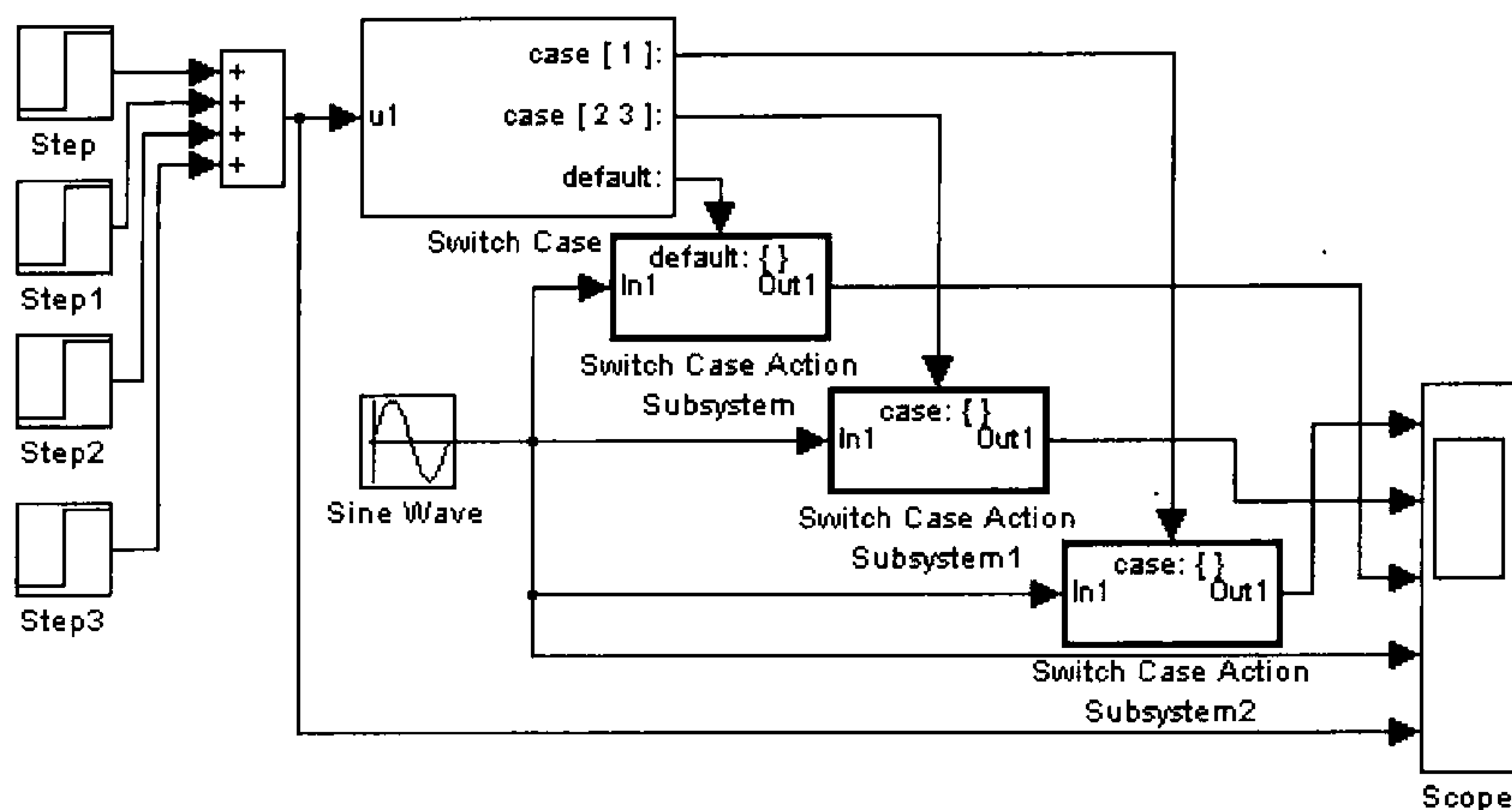


图 6-64 Switch Case 和 Switch Case Action Subsystem 子系统模型

1）对模块参数进行设置

①Step 模块参数设置。双击 Step 模块，弹出如图 6-65 所示参数设置对话框，其中“Step time”为“0”，“Initial value”为“0”，“Final value”为“1”，“Sample time”为“0”。

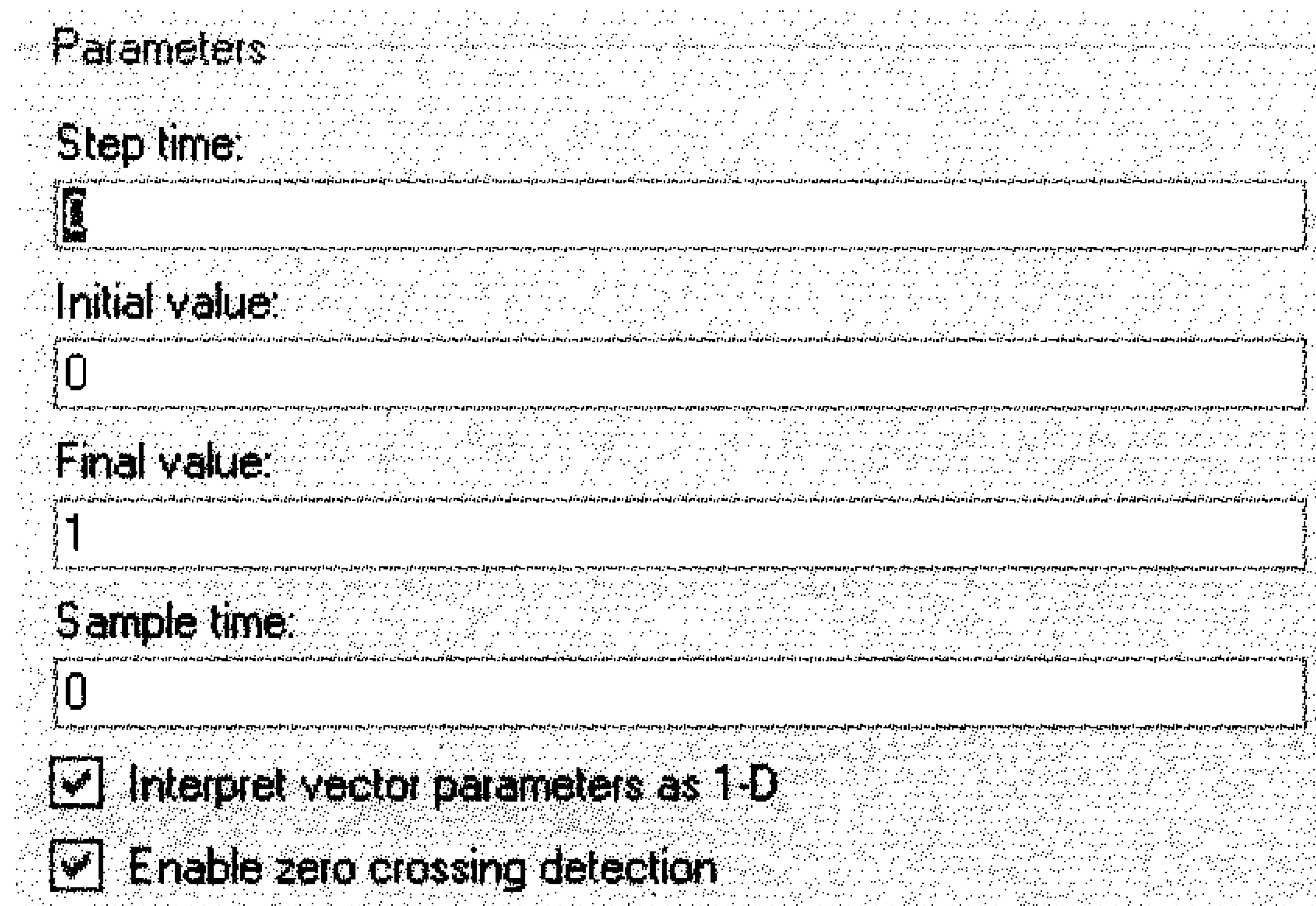


图 6-65 Step 模块参数设置

②Step1 模块参数设置。其中“Step time”为“2”，“Initial value”为“0”，“Final value”为“1”，“Sample time”为“0”。

③Step2 模块参数设置。其中“Step time”为“4”，“Initial value”为“0”，“Final value”为“1”，“Sample time”为“0”。

④Step3 模块参数设置。其中“Step time”为“6”，“Initial value”为“0”，“Final value”为“1”，“Sample time”为“0”。

⑤Sine Wave 模块参数设置。其中“Amplitude”为“1”，“Bias”为“0”，“Frequency”为“2”，“Phase”为“0”，“Sample time”为“0”。

⑥Switch Case 模块参数设置。双击 Switch Case 模块，其参数设置对话框如图 6-66 所示，其中 Case condition 设置为{1, [2, 3]}。

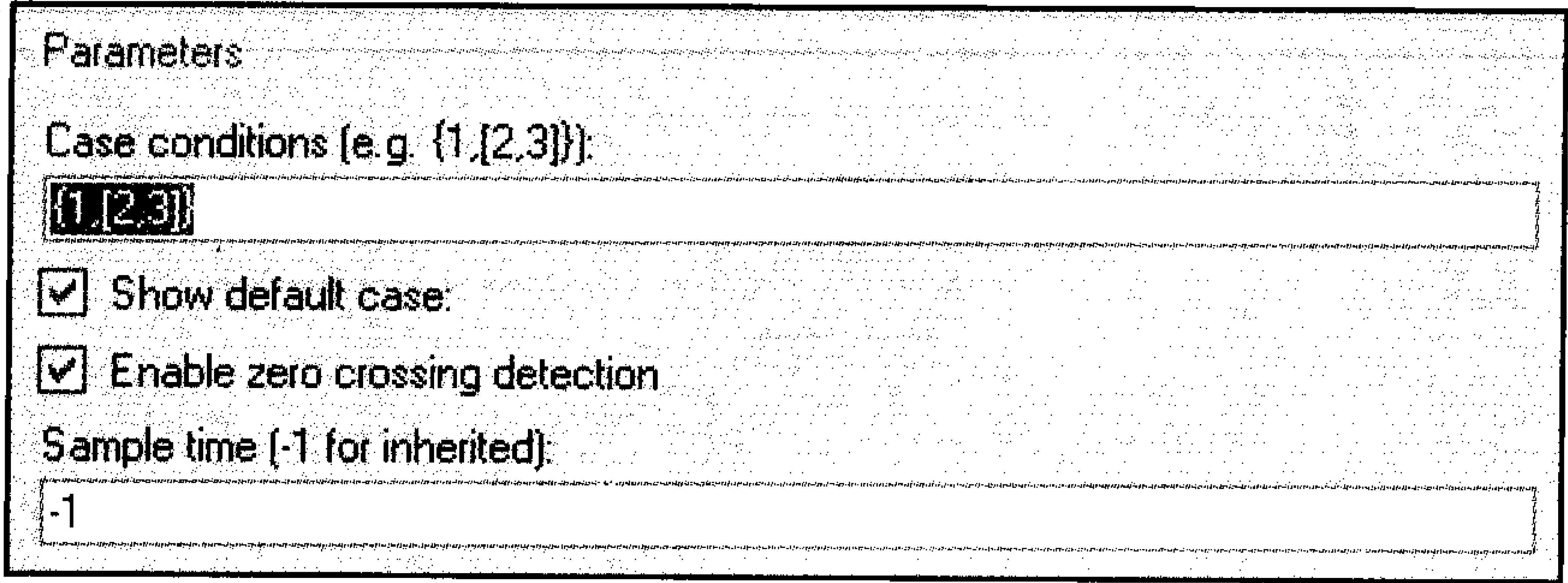


图 6-66　Switch Case 模块参数设置

⑦Switch Case Action Subsystem模块和Switch Case Action Subsystem l/2模块参数设置。在与不同的 Switch Case 模块端口连接时，"Action Port"模块显示的内容会不一样。打开 Svatch Case Action Subsystem 模块，如图 6-67 所示；打开 Switch Case Action Subsystcm l/2 模块，如图 6-68 所示。双击图 6-67 中的 Action Port 模块，弹出参数设置对话框，如图 6-69 所示，在此不改变默认参数。

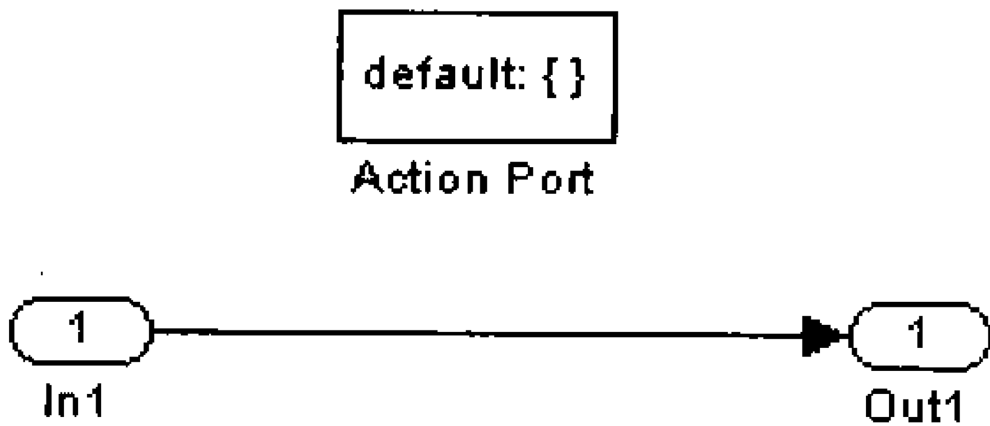


图 6-67　Switch Case Action Subsystem 模块结构

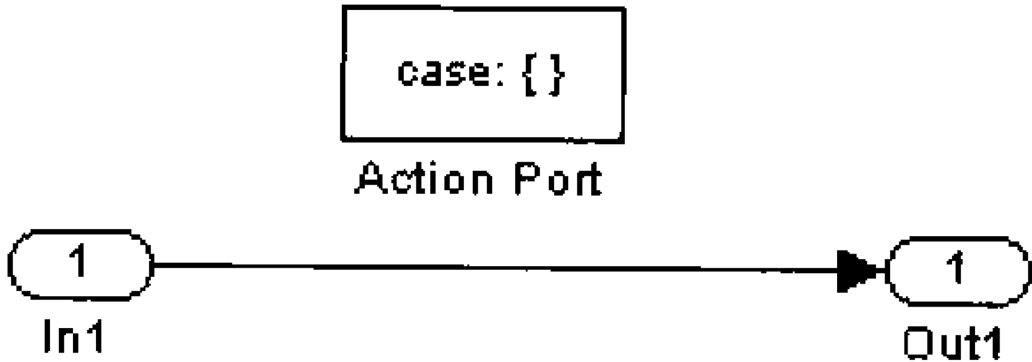


图 6-68　Switch Case Action Subsystem l/2 模块结构

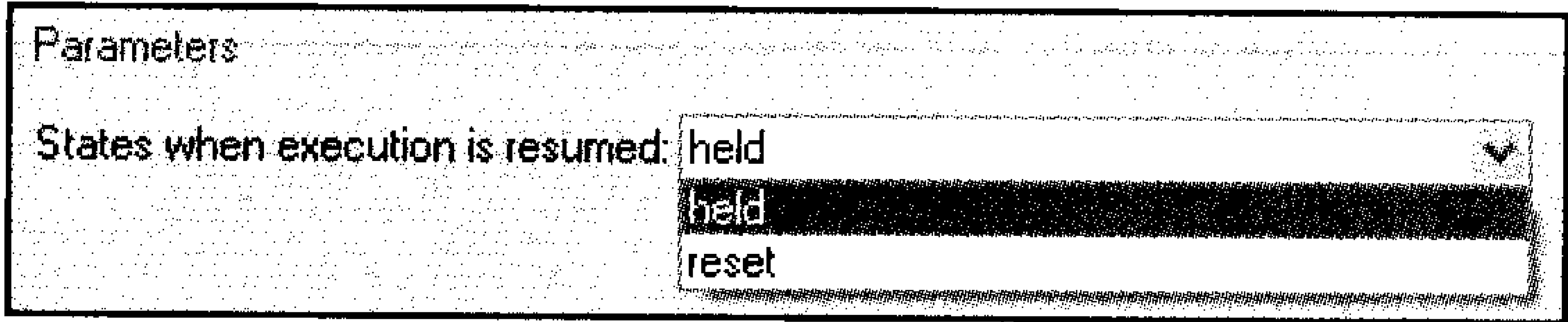


图 6-69　Action Port 模块参数对话框

2）*仿真运行及结果分析*

本例 Switch Case 和 Switch Case Action Subsystem 子系统模型的运行结果如图 6-70 所示。在该图中，第一幅子图是相对于 Switch Case 模块 Case 1 端口的结果；第二幅子图是相对于 Switch Case 模块 Case 2 端口的结果；第三幅子图是相对于 Switch Case 模块 Default 端口的结果；第四幅图是 Sine Wave 模块信号；第五幅图是 Switch Case 模块的输入信号。

据图可知，在 Switch Case 模块输入信号为 1 时，这正好对应于 Case 1 情况，则 Switch Case Action Subsystem 2 模块执行输出结果为第一幅子图；当输入信号为 2 或 3 时，Switch Case Action Subsystem 1 模块执行，输出结果为第二幅子图；当 Switch Case 模块为其他数值（如 4）时，则执行 Default 端口连接的 Switch Case Action Subsystem 模块，输出结果如第三幅子图。

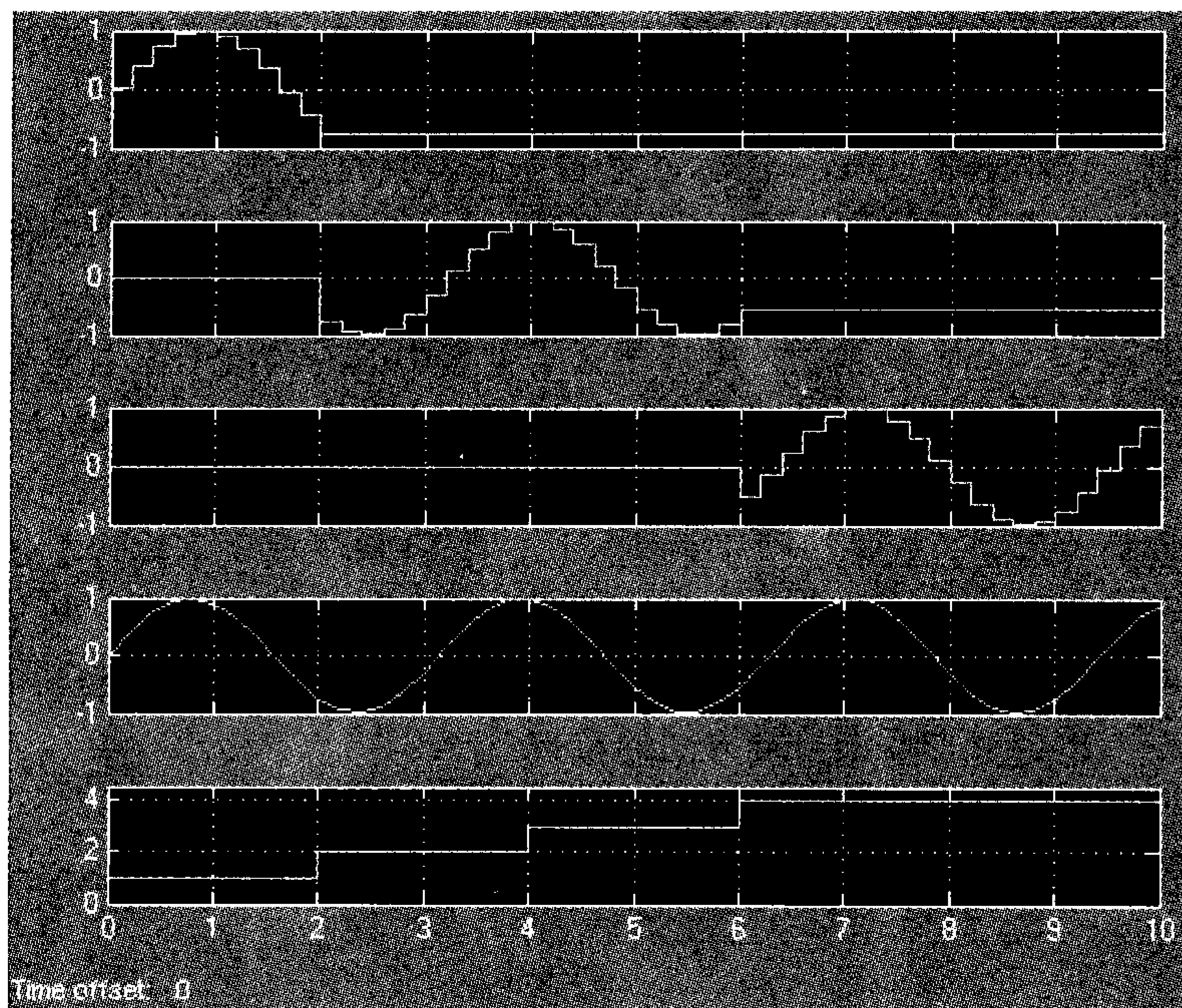


图 6-70 Switch Case 和 Switch Case Action Subsystem 子系统模型仿真结果

6.3.3 子系统封装方法与实例

封装子系统和建立子系统是两个不同的概念。建立子系统是将一组完成相关功能的模块包含到一个子系统当中，用一个模块来表示，主要是为了简化 Simulink 模型，增强 Simulink 模型的可读性，便于我们仿真和分析。在仿真前，需要打开子系统模型窗口，对其中的每个模块分别进行参数设置。虽然增强了 Simulink 模型的可读性，但并没有简化模型的参数设置。当模型中用到多个这样的子系统，并且每个子系统中模块的参数设置都不相同时，这就显得很不方便，而且容易出错；封装子系统则是将完成特定功能的相关模块集合在一起，对其中经常要设置的参数设置为变量，然后封装，使得其中变量可以在封装系统的参数设置对话框中统一进行设置。这就大大地简化了参数的设置，而且不容易出错，非常有利于进行复杂的大型系统仿真。

封装后的子系统可以作为用户的自定义模块，作为普通模块一样添加到 Simulink 模型中应用，也可添加到模块库中以供调用。封装后的子系统可以定义自己的图标、参数和帮助文档，完全与 Simulink 其他普通模块一样。双击封装子系统模块，弹出对话框，进行参数设置，如果有任何问题，可以单击【help】按钮，不过这些帮助都是要创建者自己进行编写的。

总体来说，采用封装子系统的方法有以下几点好处。

- 将子系统内众多的模块参数对话框集成为一个单独的对话框。用户可以在该对话框内输入相同子系统中不同模块的参数值。

- 可以将个别模块的描述或者帮助集成在一起，这样能有效地帮助用户了解该定制的模块（子系统）。
- 可以制作该子系统的 Icon 图标，来表示该模块的用途。
- 使用定制的参数对话框，可以避免由于不小心修改了不可改变的参数。

以上优点为模型设计带来了很大的方便，具体如下。

- 将子系统作为一个黑匣子，用户不必了解其中的具体细节而直接使用。
- 子系统中模块的参数通过对话框进行设置，十分方便。
- 保护知识产权，防止篡改。

封装的过程简单地说，就是选中子系统模块，在【Edit】菜单下执行【Mask Subsystem】命令，这时将弹出“Mask Editor”对话框，设置好这个对话框的参数，模块的封装就成功了。“Mask Editor”对话框包括 4 项内容：Icon、Parameters、Initialization 和 Documentation。这 4 项参数的设置是讲解子系统封装方法的重点。

下面，我们结合具体的实例来讲述子系统的封装方法。

例 6.8 构建简单的比例微分控制系统模型如图 6-71 所示，其 Subsystem 子系统结构如图 6-72 所示。

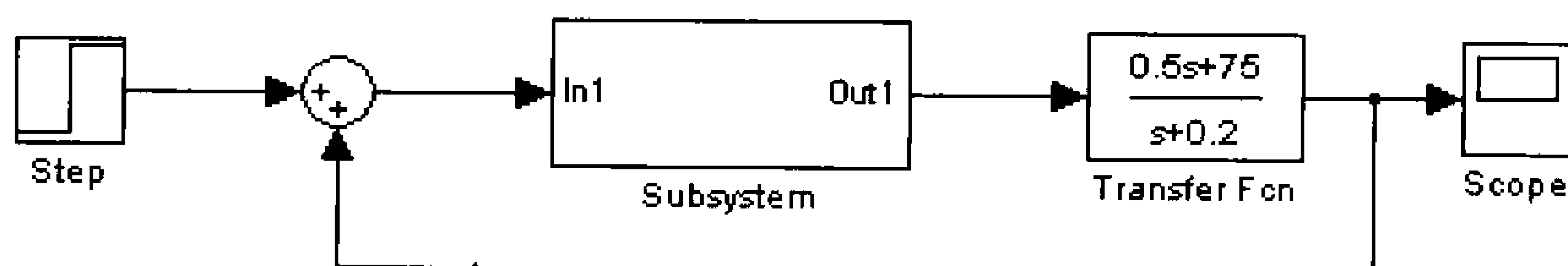


图 6-71　比例微分控制系统模型

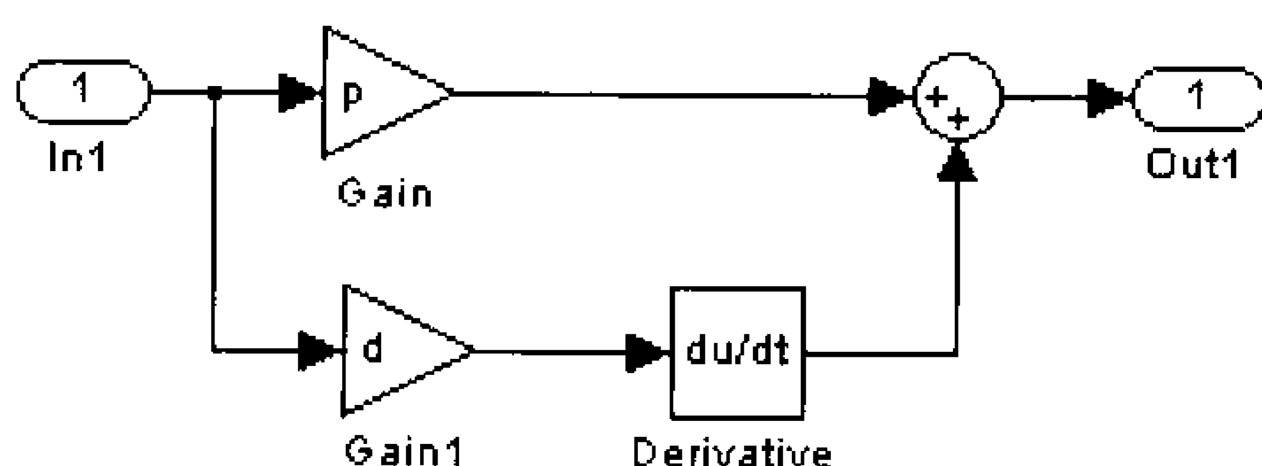


图 6-72　Subsystem 子系统结构

图 6-71 中的子系统实际上是一个比例微分控制器，其目的是通过调节其中的参数 p 和 d 实现对控制对象的有效控制。

选中图 6-70 中的 Subsystem 子系统，然后执行【Edit】→【Mask Subsystem】命令，弹出“Mask Editor”对话框，当前页面是图标（Icon）设置页面。

1．图标设置

在“Icon”中可定制封装模块的图标，如图 6-73 所示。系统提供了几种设置封装图标特性的下拉式菜单和进行个性化设置的窗口“Drawing commands”。

我们可以在模块的外观上，以最能表示模块功能的方式输入“文字”、“图像”和“转换函数”等。通过在文本框“Drawing commands”中输入命令建立用户个性化的图标，可以在图标中显示文本、图形、图像或传递函数值。

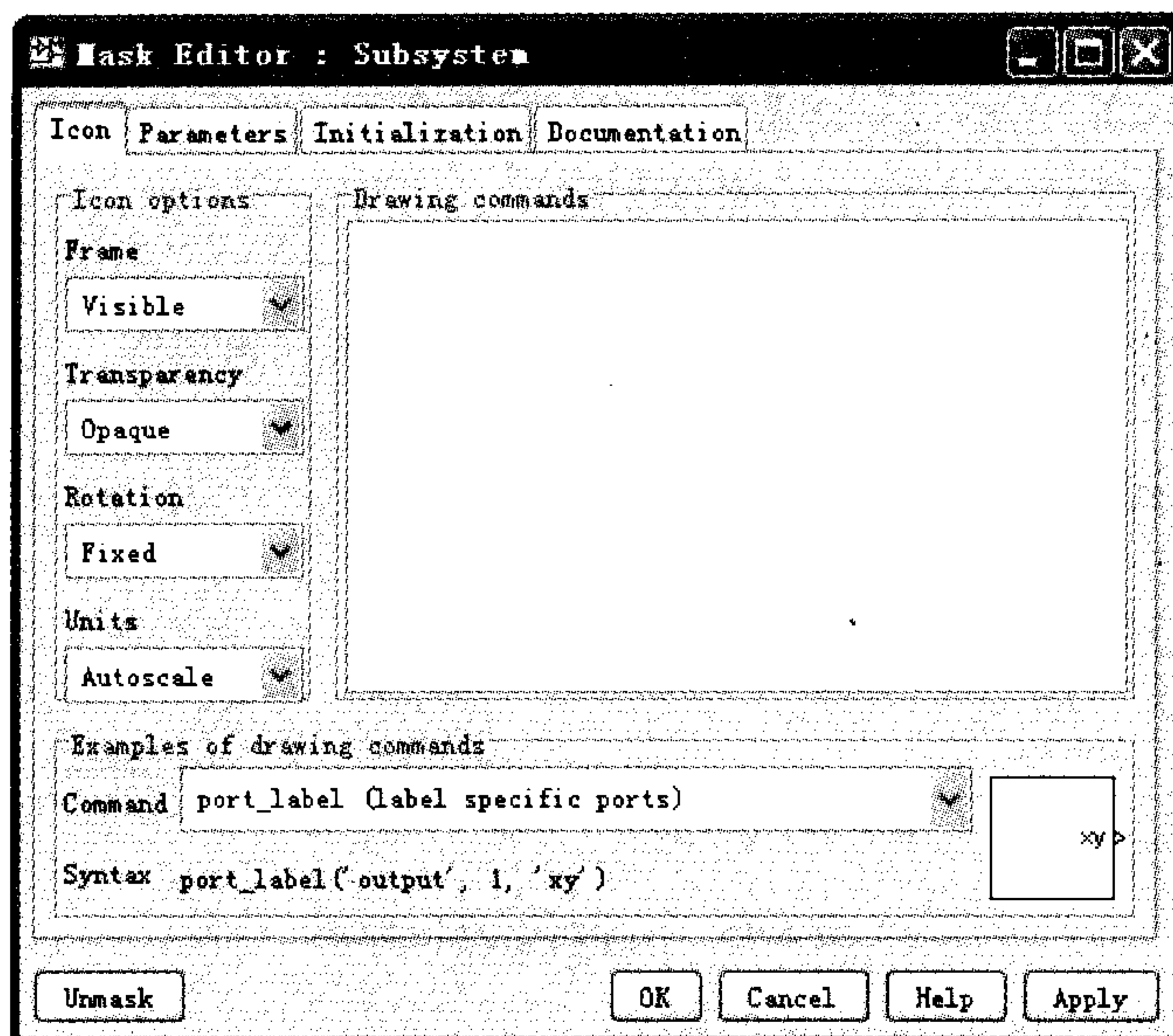


图 6-73　Icon 项参数设置对话框

1）显示文字

在对话框中显示文本的指令有以下几种。

- disp（variable/'text'）：在图标中显示变量 variable 的值或显示字符串“text”。
- text（x,y,varible/'text'）：在图标的点（*x*，*y*）处显示变量 Variable 的值或者显示字符串“text”。
- fprintf（'string'）：在图标中显示字符串。
- fprintf（'format',variable）：在图标中显示变量 variable 的值。
- port_lable（port_type,port_number,label）：根据指定的端口类型 port_type、端口号 port_number 为该端口添加标记 label。port_tpye 包括“input”和“output”，对应输入端口和输出端口；port_number 是相应端口的序号；label 是一个自己指定的字符串，用来标记相应的端口。

上述几种命令的区别在于：命令 disp 和 fpringf 把内容显示在图标的正中，而 text 命令则按照指定的位置（*x*，*y*）来显示；显示变量的值时，用 fprintf 命令可以指定值的显示格式，而命令 disp 和 text 没有该功能。命令 text 显示文本或者变量时，还可以限制文本或者变量相对于指定点（*x*，*y*）的排列方式。显示结果如图 6-74 所示。

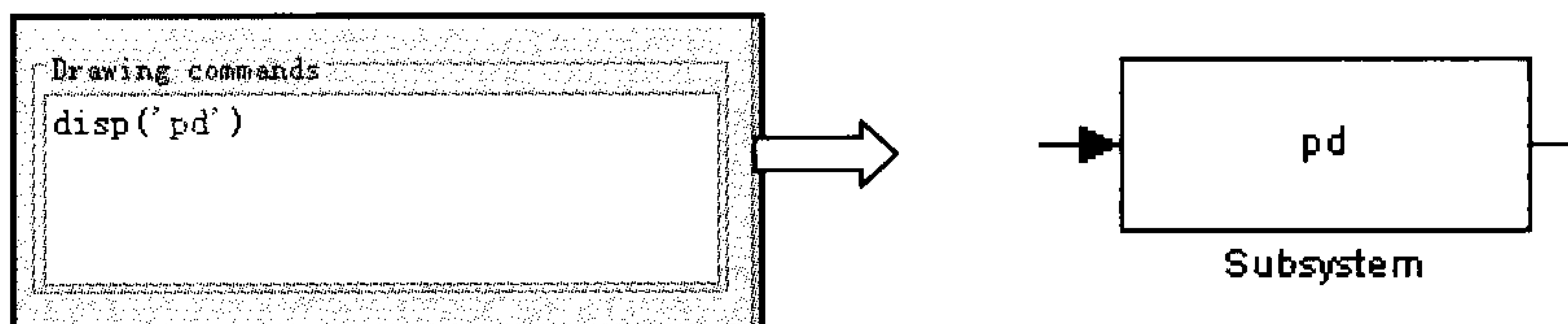


图 6-74　在图标中添加文字

2）显示图形

除了文本外，还能在封装图标中显示图形和图像。绘图命令有以下两种用法。

- plot(y)：横坐标用向量 y 中元素的序号。
- plot([$x1$, $x2$, …xn], [$y1$, $y2$, …yn])：图形会由（$x1$，$y1$）绘制出一条直线到（$x2$，$y2$），再连到（$x3$，$y3$），依次下去。



x 和 y 的数目必须是相同的，否则无法绘制出图形。

在图标上绘制图形，如图 6-75 所示。

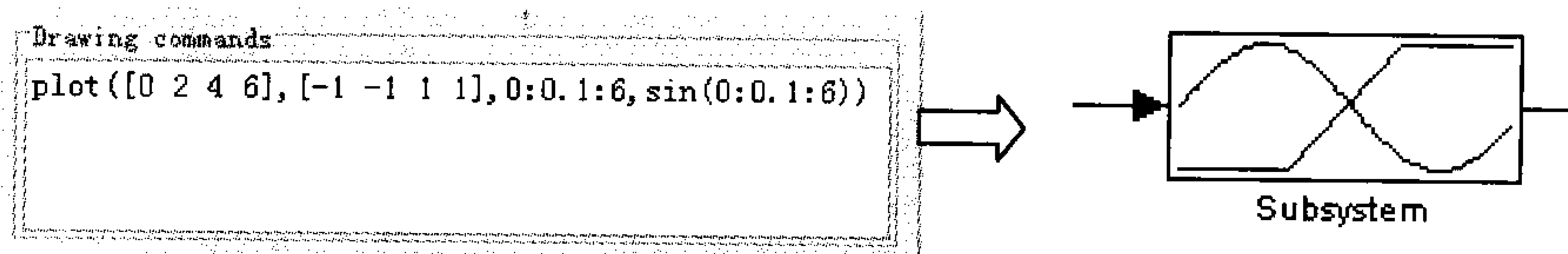


图 6-75　在图标上绘制图形

在图标上显示图像用命令 image 和 patch。

image 命令格式如下：

```
image(p)
```

这里的 p 是一个 RGB 值的三维数组。Imread('imagename')读取图形文件，并将其转换为 image 命令能够直接读取的矩阵格式。

patch 命令在曲线中填充颜色形成图像，其格式如下：

```
patch(x,y,[r g b])
```

其中 x，y 分别是曲线的横、纵坐标，[r g b]为填充颜色的 RGB 值。

3）显示转换函数

使用格式如下：

- dpoly(num,den)：num 为转换函数的分子向量，den 为转换函数的分母向量。
- dpoly(num,den,'character')：当需要显示按“z”的降幂排列的离散转换函数时 character 的值应取为“z”；当需要显示按“$1/z$”的升幂排列的离散转换函数时，character 的值应取为“z-”。
- droots(z,p,k)：显示零极点模型的转换函数，z 为零点，p 为极点，k 为增益。

4）封装图形的特性设置

使用格式如下：

- Icon frame：表示图标外的矩形边框。选择 Visible 为显示边框，Invisible 为隐藏边框。
- Icon transparency：表示图标的透明度。设置为 Opaque（不透明）时，不显示图标下的内容；设置为 Transparent 时，显示图标下的内容。
- Icon rotation：当被封装的模块旋转或翻转时，选择 Fixed 表示图标不随着转动；选择 Rotates 表示图标随着被封装的模块转动而转动。
- Icon units：这个参数用来设置 Drawing commands 中 plot 和 text 命令使用的坐标系，它包括以下 3 个选项。

- Autoscale：根据绘制的点的坐标自动选取坐标系，使得坐标中最小的 x 和最小的 y 位于图标的左下角，最大的 x 和 y 位于图标右上角。当模块大小转变时，用这种方式绘制的图形将随之发生变化。
- Normalized：规定图标左下角的坐标为（0，0），右下角的坐标为（1，1）。要绘制的点的坐标必须归一化到[0 1]之间才能显示出来。当模块大小改变时，用这种方式绘制的图标随之发生变化。
- Pixel：以像素为单位绘制图形。当模块大小改变时，用这种方式绘制的图标不会随着变化。

上面介绍了关于 Icon 参数设置的一些相关知识，对图 6-74 所示的子系统执行操作后，生成如图 6-76 所示的带图标子系统。

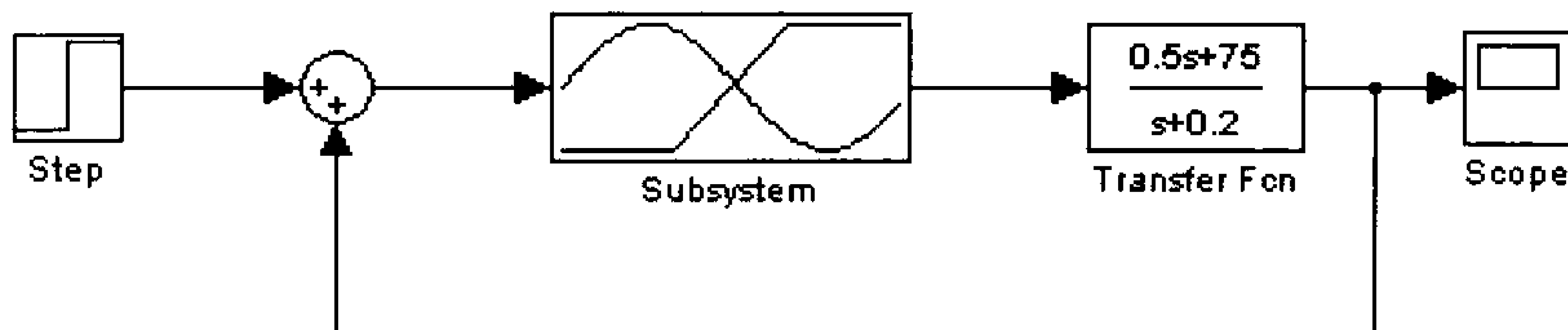


图 6-76　添加了 Icon 图标的子系统

2．**参数设置**

通过对参数设置页面的操作定义参数的“提示符（Prompt）”、“变量名（Variable）”及其他一些相关选项等，如图 6-77 所示。

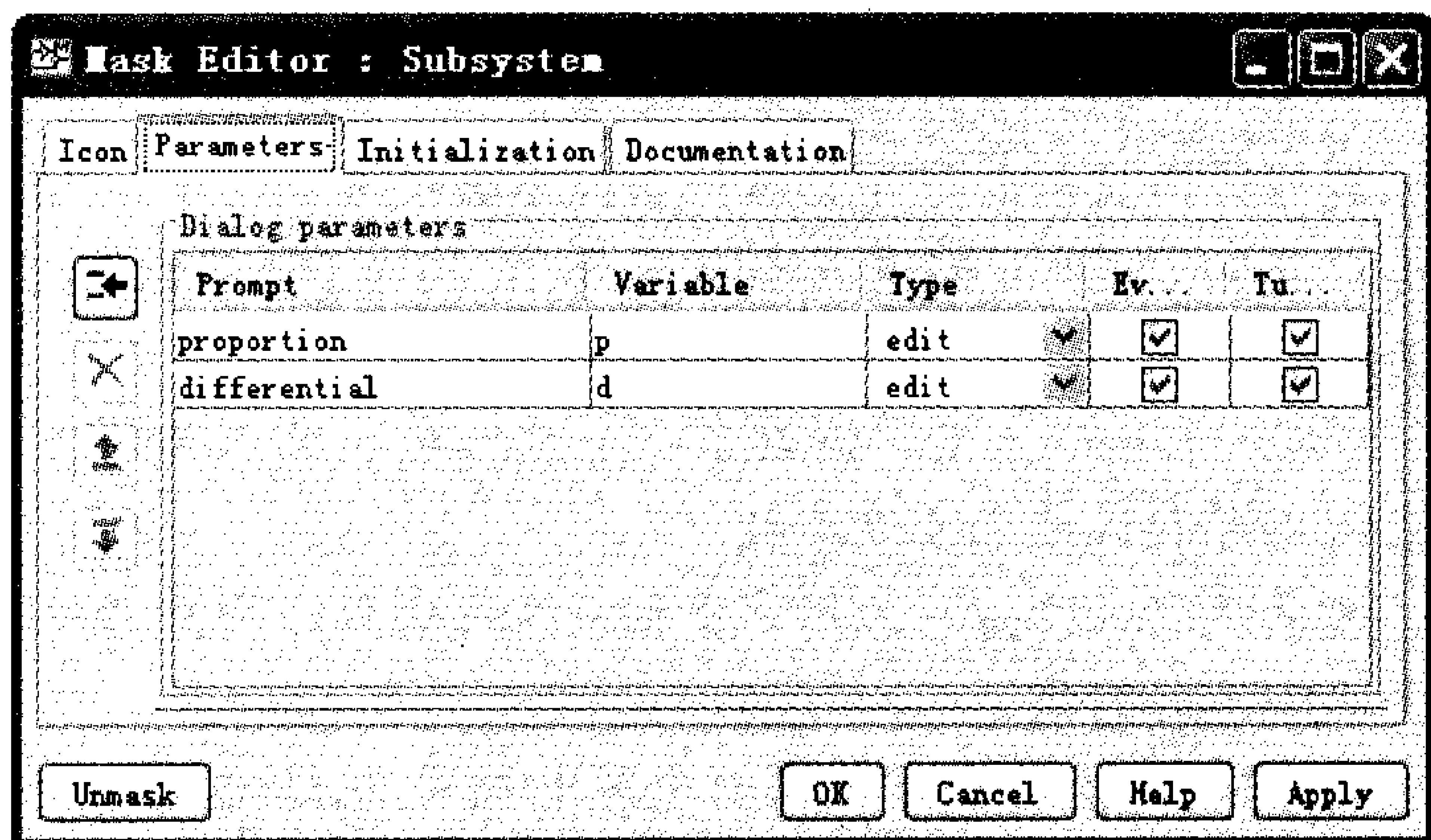


图 6-77　“Parameters”选项卡

具体操作如下所示。

- 单击图标，可在 Dialog parameters 中添加变量。
- 选中待删除的变量，单击图标，删除已添加的变量。
- 单击图标，将变量上移，在对话框中的显示位置也相应上移。
- 单击图标，将变量下移，在对话框中的显示位置也相应下移。

在编辑框“Dialog parameters”中，“Prompt”用来描述参数的文本标志，即提示符；

"Variable" 用来存储参数值的变量名；"Type" 用于选择用户的控制风格，决定在对话框中参数值是如何输入或者选中的；选中 "Evaluate" 复选项，表示用户输入的内容先由 MATLAB 进行计算，然后把结果赋值给相关变量，否则用户输入的内容不经过计算，以字符串格式直接赋给相关变量；选中 "Tunable" 复选项，允许输入值在仿真过程中发生改变。

本例中，我们要设置两个相关变量 p 和 d，设置过程如下。

（1）单击对话框中的图标；在 Prompt 后输入提示符 "比例系数"，在 Variable 后输入变量名 "p"；在 Type 栏中选择 "Edit"。

（2）再单击对话框中的图标，在 Prompt 后输入提示符 "微分系数"，在 Variable 后输入变量名 "d"；在 type 栏中选择 "Edit"。

因为变量 p 和 d 都是数值变量，因此参数均使用默认值，即勾选"Evaluate"和"Tunable"复选框。在实际使用中应设置相应的数值，子系统参数设置对话框如图 6-78 所示。

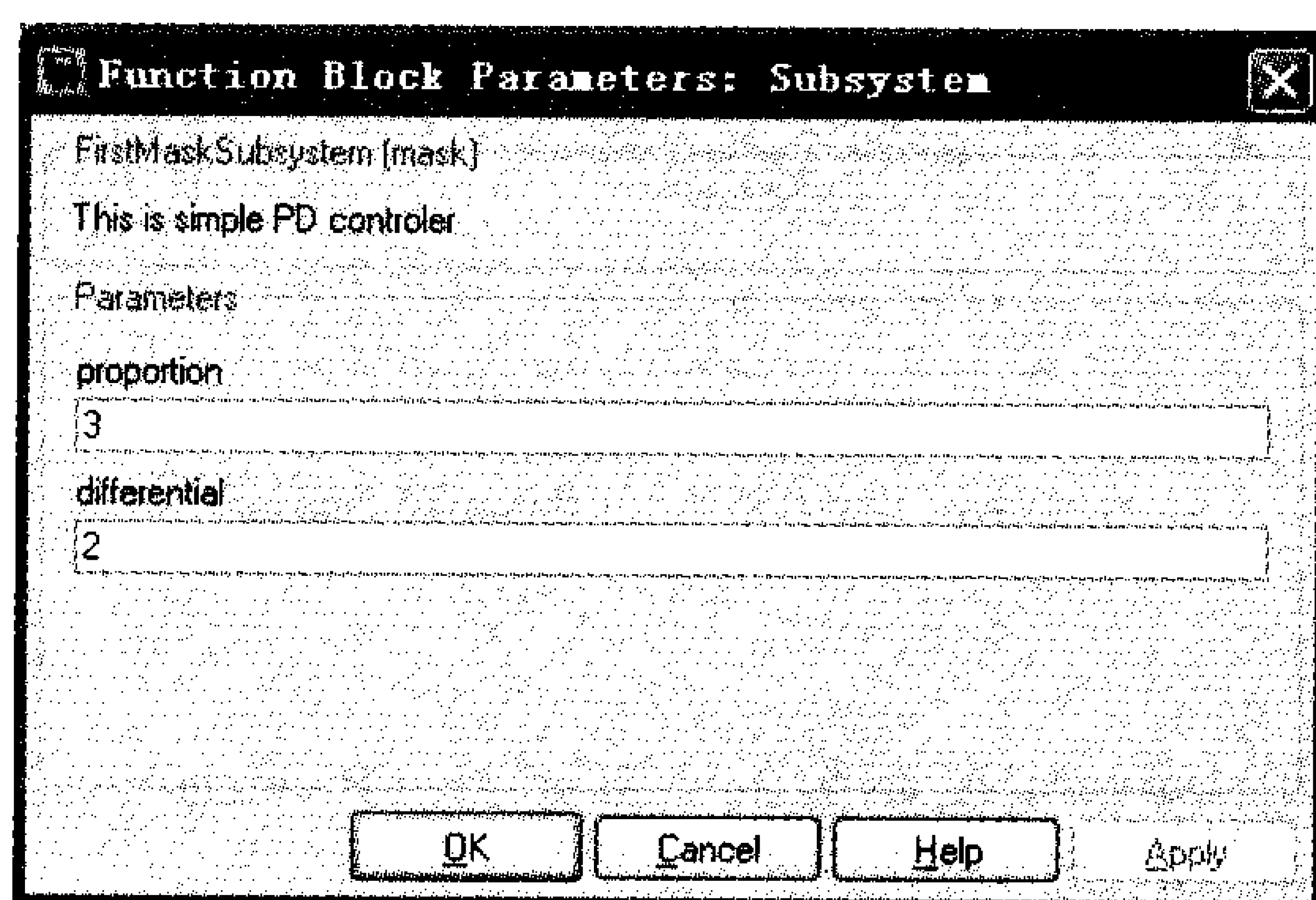


图 6-78　封装后的参数设置对话框



类型选择控制的是封装后子系统参数设置对话框，用来选择在这个对话框中提供给用户的设置参数的方式。它包括3种可选类型：Edit、Checkbox 和 Popup。

选择 "Edit" 为用户提供了一个文本框，用户通过在文本框中输入的参数值或者表达式来设置参数。

选择 "Checkbox" 为用户提供一个复选框，用户选中或者不选中复选框会返回不同的值。

选择"Popup"为用户提供一个弹出式菜单。选择 Popup 后，Popup strings 文本框被激活，在这里输入弹出菜单的选项，各个选项之间用 "|" 隔开。

选中变量后，复选框 "Show parameter" 和 "Enable parameter" 变亮，在文本框 "Dialog callback" 中可以对对话框所调用的函数进行编辑。

3．初始化设置

在 "Initialization" 页面中右方的 "Initialization commands" 文本框中可以输入初始化命令，这些命令将在开始仿真、更新模块框图、载入模型和重新绘制封装子系统的图标时被调用。所以，适当的设置有十分重要的作用，设置界面如图 6-79 所示。

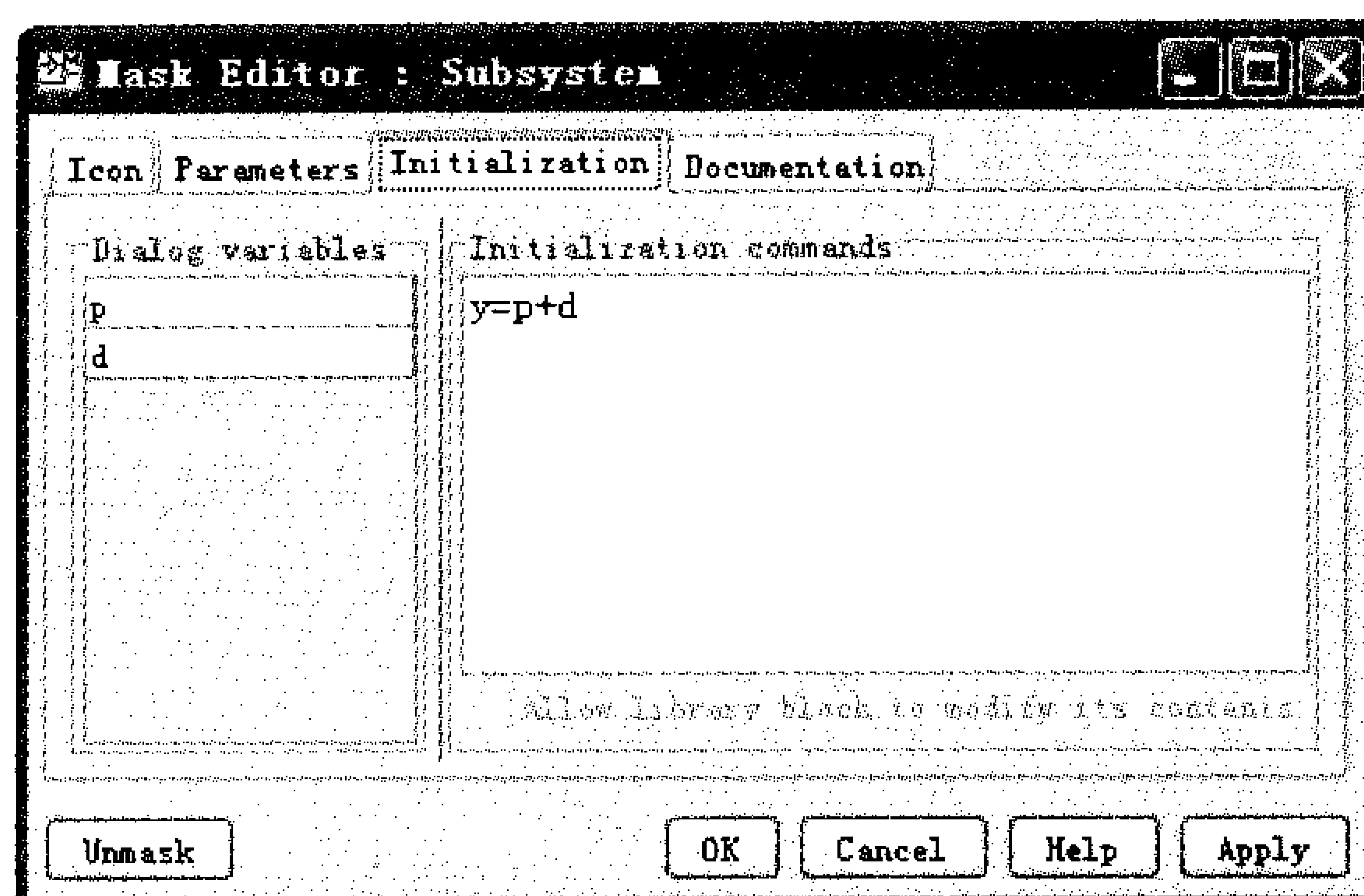


图 6-79 初始化界面设置

在左边的“Dialog variables”文本框中输入将要被初始化的变量，在右边的“Initialization commands”中进行编辑。初始化命令可以由有效的 MATLAB 表达式组成，其中包括了 MATLAB 函数、操作符和在封装工作区中定义的变量。

封装工作区指的是当封装含有初始化命令或者是封装定义了操作符及其相关变量时，Simulink 建立的一个局部工作区。封装工作区包括了和封装参数相关的变量及由初始化命令定义的变量。

初始化命令主要功能就是定义位于封装区中的变量。这些变量可以被所有被封装定义的初始化命令、封装子系统中的模块和绘制封装图标的命令使用。封装模块及其子系统内部包含的模块可以访问本封装工作区的变量，但不能访问基本工作区和其他封装工作区的变量。

在我们所举的例子中，在封装区中所定义的变量 p 和 d 是和封装子系统中的模块参数相联系的。但是，这两个变量之间不会复制给模块参数，模块是通过访问工作区中的所有变量来获取参数值的。也就是说，封装变量和模块参数之间是通过封装工作区来交互数据的。

4．封装模块的描述和帮助文本设置

现在缺少的是该模块的“说明”和“帮助”，在“Documentation”选项卡中可以定义模块的封装类型、模块描述和帮助文本，设置页面如图 6-80 所示。

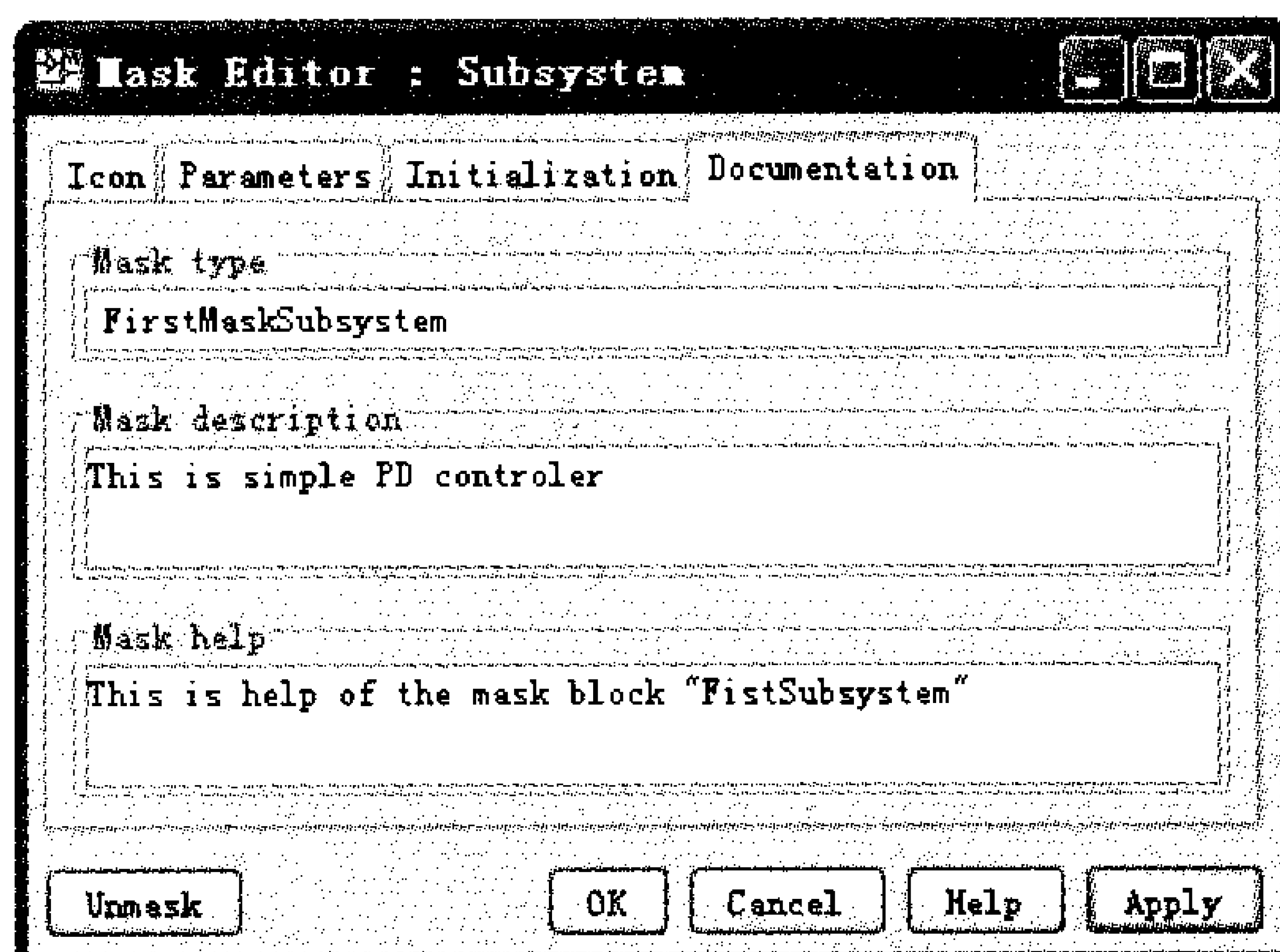


图 6-80 “Documentation”选项卡的页面设置

在文本框“Mask type”中设置模块的封装类型，没什么实际意义。可以输入字符串，其作用就是和内置的封装模块区别开来。这里输入的字符串加上“Mask”字符串显示在封装模块的对话框顶部。

在“Mask description”文本框中输入描述文本，输入的内容显示在封装模块对话框的上部，位于 Mask type 之下的边框内。输入的文本一般是对模块的目的或者功能的描述。

在“Mask help”文本框中输入文本，当单击封装模块对话框中的【Help】按钮时，就将显示这些输入的内容。

6.4 Simulink 仿真命令与回调方法

Simulink 仿真命令（简称 Simulink 命令）是一种 MATLAB 命令形式的 Simulink 建模方法。使用 Simulink 命令可以对模型和模块进行任何合法的操作，换句话说，用图形建模方法进行的操作，使用 Simulink 命令也同样可以完成。而 Simulink 命令最大的优势在于它可以以程序的方式来控制仿真模型，从而发挥 Simulink 简洁和程序控制灵活的特点。

回调也是 Simulink 的仿真命令，它在某种事件（如打开模块或删除模块等）发生时开始执行。通过回调给仿真模型提供了更强大的实用功能。

本节主要通过实例，介绍 Simulink 命令和回调的使用方法，让读者体会到命令行建模方式的优点，从而更好地使用 Simulink 进行系统建模仿真。

6.4.1 Simulink 模型构造与编辑命令

使用 Simulink 命令一样可以完成图形建模方法实现的操作，如添加/删除模块、添加/删除信号线和设置模块参数等。构造和编辑 Simulink 仿真模型的命令如表 6-15 所示。

表 6-15 构造 Simulink 仿真模型的命令

命令	功能	命令	功能
add_block	添加新模块	gcbh	获得当前模块的句柄
add_line	添加信号线	gcs	获得当前模型的路径
add_param	为模型增加参数	get_param	获得指定参数的值
bdroot	获得指定模块的最上层模型的名字	new_system	建立新的 Simulink 仿真模型或系统
close_system,bdclose	关闭模型窗口	open_system	打开一个已经存在的仿真模型或系统
delete_block	删除模块	replace_block	替代模块
delete_line	删除信号线	save_system	保存模型
delete_param	删除模型中指定的参数	Set_param	设置指定参数的值
find_system	寻找模型、模块、信号线、端口或注释	Simulink	打开 Simulink 模块库
gcb	获得当前模块的路径		

下面分别介绍这些命令的功能和调用格式，并结合实例讲述其使用方法。

1）add_block

add_block 在系统中添加模块，其调用格式如下：

- add_block('src','dest,)：复制模块 src 到 dest，这里 src 和 dest 为复制前后模块的路径名。即可以从 Simulink 的模块库中复制模块到指定的系统模型中，且复制前后模块参数完全一致。
- add_block('src','dest','parameterl',valuel....)：复制模块 src 到 dest，并重新设置复制后模块的指定参数。将参数 parameterl 的值设为 valuel。

例如，从 Simulink 模块库的子模块库 Sinks 中复制 Scope 模块到系统模型 engine 中的子系统 timing 中，其名称为 Scopel，命令如下：

```
>> add_block('Simulink / Sinks / Scope','engine / timing / Scopel')
```

从系统模型 built-in 中复制模块 Gain 到系统模型 mymodel 中，并重新改名为 Volume，参数'Gain'的值设为 4，命令如下：

```
>> add_block('buiit-in / Gain','mymodel / Volume','Gain','4')
```

2）add_line

add_line 在系统模型中添加指定的信号连线，其调用格式如下：

- h=add_line('sys','oport','iport')：在系统模型 sys 中给定模块的输出端口与指定模块的输入端口之间加入信号连线。其中，oport 和 iport 分别为输出和输入端口的路径名称（包括模块的名称和端口号）。
- h=add_line('sys','oport','iport','autorouting','on')：加入的信号线连线的方式可以由参数 autrouting 控制。'on'表示连线环绕模块，'off '表示直接以直线连接（默认）。一般环绕连接会使模型更美观一些。
- h=add_line('sys',points)：根据 points 的数据坐标在系统模型 sys 中添加一条可以不指定输入/输出端口的信号线。points 为一组坐标数据。

例如，在系统模型 mymodel 中添加从模块 Sine Wave 的输出端口 1 到模块 Mux 的输入端口 1 的信号连线，命令如下：

```
>> add_line('mymodel','Sine Wave/1','Mux/1')
```

在系统模型 mymodel 中添加一条从坐标（20，55）到（40，10）到（60，60）的信号连线，命令如下：

```
>> 'add_line('mymodel',[20 55; 40 10; 60 60])
```

3）add_param

add_param 为系统模型增加指定参数并赋值，其调用格式如下：

add_param('sys','parameterl',valuel',,'parameter2',value2,⋯)：该命令为系统模型增加系统参数，并初始化参数值以供仿真时的相关模块来调用。当参数被加入到系统模型中后，set_param 和 get_param 这两个命令就能够像对系统本身的 Simulink 参数一样来对新加入的参数进行操作。

例如，为系统模型 vdp 增加参数 paraml 和 param2，并分别赋值为 valuel 和 value2，命令如下：

```
>> add_param('vdp','paraml','valuel','param2','value2')
```

4）bdroot

bdroot 返回当前系统或模块的顶层系统模型的名称，或指定所选系统或模块的顶层系统模型的名称，其调用格式如下：

- bdroot：返回当前系统或模块的顶层系统模型的名称。

- bdroot('obj')：指定所选系统或模块的顶层系统模型的名称。

5）close_system，bdclose

close_system 和 bdclose 关闭一个系统模型或一个模块对话框，其调用格式如下：

- close_system：无参数，表示关闭当前的系统或子系统。如果当前系统为最顶层，且该系统经过修改，则执行命令后系统会提示是否保存系统模型。
- close_system('sys')：关闭指定的'sys'系统或子系统模型窗口。
- close_system('sys'，saveflag)：关闭指定的顶层系统模型窗口，并从内存中清除。当 saveflag=0 时，关闭模型时不保存；当 saveflag=1 时，关闭时使用当前名字保存系统模型。
- close_system('sys'，'newname')：将指定的系统模型保存至新的名为'newname'的模型文件中并关闭该系统模型。
- close_system('blk')：关闭与模块'blk'相关联的对话框。
- bdclose：无条件关闭当前系统模型，并不保存修改。
- bdclose('sys')：无条件关闭指定的系统模型。
- bdclose('all')：无条件关闭所有系统模型。

例如，关闭当前系统或子系统，命令如下：

```
>> Close_system
```

关闭并保存系统模型'engine'，命令如下：

```
>> close_system('engine', 1)
```

关闭系统模型'engine'下 Combustion 子系统中的 Unit Delay 模块对话框，命令如下：

```
>> close_system('engine / Combustion / Unit Delay')
```

无条件关闭所有系统模型，不保存修改内容，命令如下：

```
>> bdclOSe('all')
```

6）delete_block

delete_block 删除指定模块，其调用格式如下：

delete_block('blk')：从系统模型中删除由 blk 指定的模块，此处 blk 为该模块的完整路径名。

例如，从系统模型 vdp 中删除模块 Outl，命令如下：

```
>> delete_block('vdp / Outl')
```

7）replace_block

replace_block 替代系统中的指定模块，其调用格式如下：

- replace_block('sys','blkl','blk2','noprompt')：使用 blk2 替代系统模型 sys 中的模块 blkl，如果 blk2 为 Simulink 的内置模块，则只需要给出模块的名称即可，如果为其他模块，必须给出该模块的完整路径名。
- replace_block('sys','parameter','value','blk',...)：使用 blk 取代模型 sys 中具有特定参数取值的所有模块。Parameter 为参数名，value 为其取值。可以指定任何数量的参数取值来作为替换条件。

例如，使用 Integrator 积分模块替代系统模型 f14 中的 Gain 增益模块，并将被替换的模块路径名保存到 RepNames 中。在替换之前，Simulink 会给出匹配的模块列表，用户可以选择进行替换。如果用户不希望弹出列表框，则可以使用 noprompt 参数来屏蔽，命令如下：

```
>> RepNames=replace_block('f14','Gain','Integrator')
```

或者

```
>> RepNames=replace_block('f14','Gain','Integrator','noprompt')
```

运行后，RepNames='f14/Gain'。

使用 Integrator 积分模块替代 clutch 系统中参数 Gain 的值为变量 bv 的模块，命令如下：

```
>> replace_block('clutch','Gain','bv','Integrator')
```

注意 使用该命令对系统的修改，是无法使用 undo 命令来撤销的。所以，在使用该命令前建议先保存模型。

8）delete_line

delete_line 删除系统模型的指定信号连线，其调用格式如下：

- delete_line('sys','oport','iport')：删除系统模型 sys 中的指定信号连线。其中，oport 和 iport 分别为输出和输入端口的路径名称。
- delete_line('sys',[x y])：删除系统模型 sys 中的一条通过点（x，y）的信号连线。

例如，在系统模型 mymodel 中删除从模块 Sine Wave 的输出端口 1 到模块 Mux 的输入端口 2 的信号连线，命令如下：

```
delete_line('mymodel','Sine Wave/1','Mux/2')
```

9）get_param

get_param 获得指定系统模型或模块的指定参数的值，其调用格式如下：

- get_param('obj','parameter')：返回由 obj 指定的系统模型或模块的参数 parameter 的值。
- get_param({objects},'parameter')：返回多个模块指定参数的取值，其中{objects}表示模块的元胞数组（Cell）。
- get_param(handle,'parameter')：返回句柄值为 handle 的对象的指定参数的取值。
- get_param(0,'parameter')：返回 Simulink 当前的仿真参数或默认模型、模块的指定参数的取值。
- get_param('obj','ObjectParameters')：返回描述某一对象参数取值的结构变量。其中返回到结构变量中的每一个参数域分别包括参数名称、数据类型以及参数属性等。
- get_param('obj','DialogParameters')：返回指定模块对话框中所包含的参数名称的元胞数组。

例如，获得系统模型 exap 中 Gain 模块的参数 Gain 的值，命令如下：

```
>> get_param('exap / Gain','Gain')
ans =
    1
```

获得当前选中模块的名称，命令如下：

```
>> get_param(gcb,'Name')
```

获得正弦信号模块 Sine Wave 的控制对话框中所包含的参数名称，命令如下：

```
>> p=get_param('untitled / Sine Wave','DialogParameters')
```

结果如下：

```
P =
```

```
'Amplitude'
'Frequency'
'Phase'
'SampleTime'
```

10）delete_param

delete_param 删除系统模型中由 add_param 添加的参数，其调用格式如下：

delete_param('sys','parameterl','arameter2',...)：删除系统模型中由add_param添加的参数。如果要删除的参数不是由 add_param 添加的，则 Simulink 会提示错误信息。

例如，删除系统模型 vdp 中的参数 paraml，命令如下：

```
>> delete_param('vdp','paraml')
```

如果 paraml 为内置参数，则会有下面的错误信息：

```
>> parameter'paraml'is a buiit-in parameter and can not be deleted.
```

11）find_system

find_system 寻找模型、模块、信号线、端口或注释，其调用格式如下：

find_system(sys,'cl',cvl,'c2',cv2,…'p1',v1,'p2',v2,…)：查找由模型名称 sys 和各条件指定的系统模型、模块、信号线或注释等，并返回相应的路径名与操作句柄。sys 可以是一个路径名、包含路径的元胞数组或操作句柄。

例如，查找当前所有已打开的系统和模块，并返回包含它们名称的元胞数组，命令如下：

```
>> find_system
```

查找 vdp 系统中所有增益参数为 1 的增益模块的名称，命令如下：

```
>> gb=find_system('vdp','BlockType','Gain')
>> find_system(gb,'Gain','l')
```

下面这条语句与上面两条功能一致：

```
>> find_system('vdp','BlockType','Gain','Gain','1')
```

获得 vdp 系统中所有的信号线和注释的操作句柄，命令如下：

```
>> sys=get_param('vdp','Handle');
>> l=find_system(sys,'FindAll','on','type','line');
>> a=find_system(sys,'FindAll','on','type','annotation');
```

12）gcb

gcb 返回当前模块的路径名，其调用格式如下：

- gcb：返回当前模块的路径名。
- gcb('sys')：返回指定系统的路径名。

例如，获得最近单击的模块的路径名，命令如下：

```
>> gcb
ans=
          exap / Gain
```

13）gcbh

gcbh 返回当前模块的操作句柄，其调用格式如下：

gcbh：返回当前模块的操作句柄。

14）gcs

gcs 返回当前系统的路径，其调用格式如下：

gcs：获得当前系统的路径名。

15）new_system

new_system 建立新的 Simulink 仿真模型或系统，其调用格式如下：

new_system('sys')：使用参数给定的名称建立一个新的 Simulink 系统模型。如果'sys'为一个路径，则新建立的系统为在此路径中指定的系统模型下的一个子系统。注意，new_system 命令并不打开系统模型窗口。

例如，新建一个名为'mysys'的系统模型，命令如下：

```
>> new_system('mysys')
```

在系统模型 vdp 下新建一个名为'mysys’的子系统，命令如下：

```
>> new_system('vdp/mysys’)
```

16）open_system

open_system 打开一个已存在的系统模型窗口或模块对话框，其调用格式如下：

- open_system('sys')：打开一个指定的系统模型或子系统的窗口。这里的'sys'使用的是 Matlab 标准的路径名，指定一个系统模型的绝对路径，或相对于已经打开的系统模型的相对路径名。
- open_system('blk')：打开由'blk'指定的模块对话框。'blk'为一个模块的绝对路径名。如果该模块的回调函数 OpenFcn 被指定了例程，则该例程也被执行。
- open_system('blk','force')：打开封装后的子系统，'blk'为封装子系统模块的路径名。该命令与封装子系统右键菜单中的 Look under mask 的功能一致。

例如，打开名为'controller'的系统模型，命令如下：

```
>> open_system('controller')
```

打开 controller 模型下的增益模块 Gain 的对话框，命令如下：

```
>> open_system('controller / Gain')
```

打开 controller 模型下的封装子系统'subsystem'，命令如下：

```
>> open_system('controller / subsystem'，'force')
```

17）save_system

save_system 保存一个 Simulink 系统模型，其调用格式如下：

- Save_system：使用当前名称保存当前顶层的系统模型。
- save_system('sys')：保存由'sys'指定的系统模型。
- Save_system('sys','newname')：使用新名称'newname'保存由'sys'指定的系统模型。

例如，保存当前系统模型，命令如下：

```
>> save_systerm
```

保存系统模型 vdp，命令如下：

```
>> save_system('vdp')
```

将系统模型 vdp 另存为 myvdp，命令如下：

```
>> save_system('vdp','myvdp')
```

18）set_param

set_param 设置系统模型或模块指定参数的值，其调用格式如下：

set_param('obj','parameterl',valuel,'parameter2',value2,...)：其中，obj 表示系统模型或其中的系统模块的路径名称，或者取值为 0；该命令用于给指定的参数设置合适的值，取值

为 0 表示将指定的参数设置为默认值。在仿真过程中，使用此命令可以在 Matlab 的工作区间中改变这些参数的取值，从而可以更新系统在不同的参数下运行仿真。

例如，设置系统模型 vdp 的求解器为 ode15s，仿真结束时间为 3000s，命令如下：

```
>> set_param('vdp','Solver','ode15s','StopTime','3000')
```

设置系统模型 exap 中的增益模块的参数 Gain 的值为 10，命令如下：

```
>> set_param('exap/Gain','Gain','10')
```

设置系统模型 vdp 中 Fcn 模块的位置为[50 100 110 120]，命令如下：

```
>> set_param('vdp/Fcn','Position',[50 100 110 120])
```

19）Simulink

Simulink 打开 Simulink 模块库浏览器。

在实际应用中，纯粹使用上述命令行的方式来构造和编辑模型是既麻烦且低效的，它更多的是起到辅助图形构造和编辑的作用，当高级用户对所建立的动态系统模型进行系统仿真与分析时，为其提供更为灵活的控制方式。

6.4.2 Simulink 模型仿真命令

在 Simulink 图形窗口仿真方式下，往往不能定义仿真模型的执行次数，也就是说通过图形窗口方式只能一次一次地手动执行来达到重复仿真的目的。这在模型性能比较时很不方便。然而，通过模型仿真命令可以实现重复仿真的目的，同时还可以改变模型的参数，控制模型的运行方式，方便了模型的分析与仿真。下面将逐一介绍 Simulink 模型仿真命令。

1）sim 命令

sim 命令可以使用户在 MATLAB 命令窗口或者 M 文件中运行由 Simulink 建立的模型。这无疑大大方便了模型的分析和仿真，也进一步丰富了分析和仿真的内容，比如研究不同的参数、输入和初始条件的影响等。

sim 命令有以下 3 种调用格式：

- [*t*,*x*,*y*]=sim('modelname')：利用对话框参数进行仿真，返回输出矩阵。
- [*t*,*x*,*y*]=sim('modelname',timespan,options,ut)：利用输入参数进行仿真，返回输出矩阵。
- [*t*,*x*,*y*1,*y*2,…,yn]=sim('modelname',timespan,options,ut)：利用输入参数进行仿真，返回逐个输出。

现对上面调用格式中的参数说明如下：

- *t*：返回仿真的时间向量。
- *x*：返回仿真的状态矩阵，前面是连续状态，接着是离散状态，它的每个列表示一个“状态”变量的记录。
- *y*：仿真的输出矩阵，它取自模型中输出口模块记录，每一列对应于一个输出端口（Outport）的值。
- modelname：被运行的模型名（不含扩展名）。模型文件必须在 MATLAB 搜索路径上。
- timespan：时间带宽，用来指定仿真的时间区间。它有以下 4 种形式。

- []：“空”，利用模型对话框设置时间。
- [tFinal]：标量，指定仿真停止时间，仿真开始时间默认为 0。
- [tStart tFinal]：二元向量，指定仿真区间。
- [tStart OutputTimes tFinal]：指定开始时间、停止时间和要输出的时间点。

- options：是在 MATLAB 特定的一种数据结构。它是由 simset 命令产生的一个结构体用来指定仿真的参数。在设置仿真参数上，它具有最高的优先权，可以覆盖模型参数设置框中的设置。
- ut：向顶层输入端口模块输入的外部数据，将值赋给仿真对象的输入端口模块。ut 可以是 MATLAB 的函数（以字符串的形式表达），用来指定每一个时间点的输入 u=UT(t)。这是个包含所有输入端口的输入值表，或者用逗号分隔的列表 ut1，ut2，…，每一个都对应于一个特定的端口。

例如，使用参数仿真系统 vdp，命令如下：

```
>> [t,x,y] = sim('vdp')
```

设置参数'Refine'的值，并仿真系统 vdp，命令如下：

```
>> [t,x,y] = sim('vdp',[],simset('Refine',2))
```

设置仿真时间为 1000s，并设置多个参数，命令如下：

```
>> [t,x,y] = sim('vdp',1000,simset('MaxRows',100, 'Output Variables', 'ty', 'FinalStateName', 'xFinal'))
```

2）simset 命令

simset 命令是用来向 sim 命令产生或者编辑仿真参数和积分器属性的命令，设置后，MATLAB 会建立一个名为“Options”的结构变量，用来保存用户设置的参数值。它有以下 4 种调用格式：

- options=simset(property,value,...)：把“property”的参数赋值为“value”，结果保存在 options 中。
- options=simset(old_opstruct,property,value,...)：把已经存在的结构“old_opstruct”的参数“property”重新赋值为“value”，结果保存在新结构 options 中。
- options=simset(old_opstruct,new_opstruct)：用结构“new_opsturct”的值代替已经存在的结构“old_opstruct”的值。
- simset 不带参数：显示所有的参数名和它们可能的值。

例如，创建名为 myopts 的 options 结构体变量，设置 MaxDataPoints 的值为 100，并使用该变量对系统模型进行设置和仿真。

```
>> myopts = simset('MaxDataPoints',100,'Refine',2)
>> [t,x,y] = simset('vdp',10,myopts)
```

3）simplot 命令

simplot 命令在窗口中绘制仿真数据图形，类似 scope 模块，其调用格式如下：

- simplot(data)
- simplot(time,data)

其中，data 表示动态系统仿真结果的输出数据（一般由 Output、To Workspace 等模块产生）。其数据类型可以为矩阵、向量或结构数组。time 为动态系统仿真结果的输出时间向量。当系统输出数据为带有时间向量的结构数组时，此参数被忽略。

例 6.9 建立如图 6-81 所示的正弦信号 Simulink 模型 exa07_09，其中 To Workspace 模

块的名称为 yout，利用仿真命令对模型进行仿真，并绘制仿真数据图形。

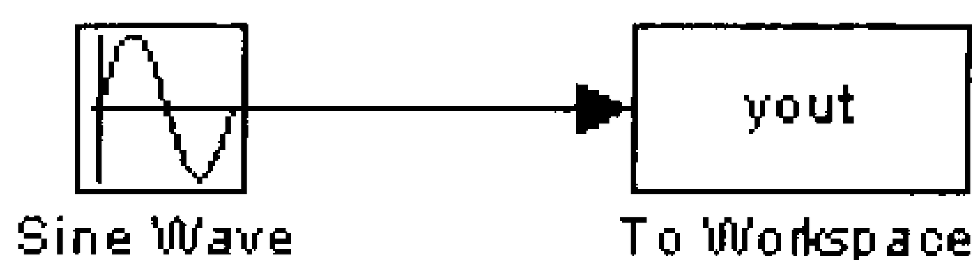


图 6-81　正弦信号 Simulink 模型

在保证 exa07_09 模型所在路径为 MATLAB 当前工作路径的前提下，在 MATLAB 命令窗口中输入：

```
>> sim('exa07_09')
```

使 exa07_09 模型运行，仿真结束后，输入命令：

```
>> simplot(tout,yout)
```

绘制的模型输出数据图形如图 6-82 所示。

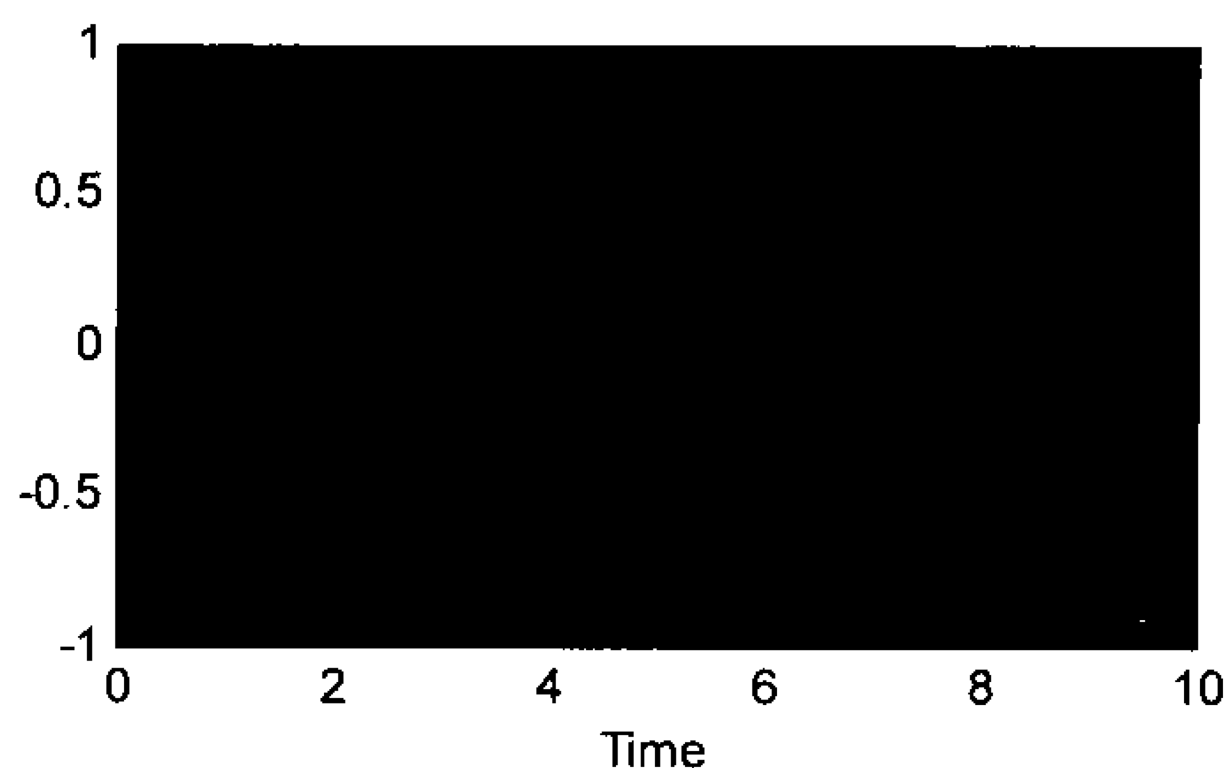


图 6-82　simplot 绘制的模型输出数据图形

4）simget 命令

simget 命令用来获取仿真模型的参数设置值，其调用格式如下：

- struct=simget(modelname)：返回指定模型 model 的参数设置的 options 结构。
- value=simget(modelname,property)：返回指定模型 model 参数 property 的值。
- value=simget(OptionStructure,property)：获取参数设置的 OptionsStructure 中参数 property 的值。

例如，获得模型 exa07_09 的所有仿真参数设置的结构体变量，命令如下：

```
>> options = simget('exa07_09')
```

得到的输出结果如下：

```
options =
                        AbsTol: 'auto'
                         Debug: 'off'
                    Decimation: 1
                  DstWorkspace: 'current'
                FinalStateName: ''
                     FixedStep: 'auto'
                  InitialState: []
                   InitialStep: 'auto'
                      MaxOrder: 5
      ConsecutiveZCsStepRelTol: 2.8422e-013
```

```
      MaxConsecutiveZCs: 1000
             SaveFormat: 'Array'
          MaxDataPoints: 1000
                MaxStep: 'auto'
                MinStep: 'auto'
  MaxConsecutiveMinStep: 1
           OutputPoints: 'all'
        OutputVariables: 'ty'
                 Refine: 1
                 RelTol: 1.0000e-003
                 Solver: 'ode45'
           SrcWorkspace: 'base'
                  Trace: ''
              ZeroCross: 'on'
          SignalLogging: 'on'
      SignalLoggingName: 'logsout'
     ExtrapolationOrder: 4
 NumberNewtonIterations: 1
                TimeOut: []
```

6.4.3　模型与模块的回调方法

一个模型或者模块在建模或仿真过程中都有相应的事件，比如加载、删除和执行等，可以通过回调来设置这些事件发生时所要执行的命令。也就是说，回调可以使仿真模型变得更加灵活实用。

1．回调函数介绍

Simulink 的回调可以应用于模型或模型中某个指定的模块。模型和模块的回调函数分别如表 6-16 和表 6-17 所示。

表 6-16　模型回调函数

回调函数	功　能	回调函数	功　能
CloseFcn	设置模型关闭时的响应事件	PreSaveFcn	设置模型保存前的响应事件
PostLoadFcn	设置模型加载后的响应事件	StartFcn	设置模型仿真开始前的响应事件
PostSaveFcn	设置模型保存后的响应事件	StopFcn	设置模型仿真结束后的响应事件
PreLoadFcn	设置模型加载前的响应事件		

表 6-17　模块回调函数

回调函数	功　能	回调函数	功　能
CloseFcn	设置模块由 close_system 命令关闭时的响应事件	ParentCloseFcn	设置包含当前模块的子系统关闭前的响应事件
CopytFcn	设置模块被复制时的响应事件	PreSaveFcn	设置模型保存前的响应事件，适用于子系统
DeletetFcn	设置模块被删除时的响应事件，适用于子系统	PostSaveFcn	设置模型保存后的响应事件，适用于子系统
InitFcn	设置模块被编译及赋值前的响应事件	StartFcn	设置模块编译后而仿真开始前的响应事件

（续表）

回调函数	功能	回调函数	功能
LoadFcn	设置模块被加载后的响应事件，适用于子系统	StopFcn	设置仿真在任何方式停止时的响应事件
ModelCloserFcn	设置模型关闭前的响应事件，适用于子系统	UndoDelete	设置撤销模块删除操作时的响应事件
OpenFcn	设置模块被打开时的响应事件		

2．加载回调函数

加载模型或模块的回调函数有两种方法：使用 MATLAB 的 set_param 命令，或者使用模型或模块的 Callbacks 对话框。

1）使用 set_param 命令

使用 set_param 命令加载回调函数的具体格式如下：

set_param(Object,parameter,value)

现对上面的参数描述如下：

object 为包含模型名或模块路径。如果回调是关于模型动作的，则 object 为模型名；如果回调是关于模块动作的，则 object 为该模块的 Simulink 路径名。

parameter 是一个包含表 6-16 和表 6-17 中模型或模块的回调函数名。

value 是一条 MATLAB 命令或包含 MATLAB 命令的 M 文件名称。当 Parameter 所指定的事件发生时，则运行 value 所指定的 MATLAB 命令。

2）使用模型或模块的 Callbacks 对话框

可以使用 Callbacks 对话框来代替 set_param 命令来加载回调函数。

打开模型，执行【File】→【Model properties】命令，打开模型属性对话框，单击“Callbacks”选项卡，则出现模型的 Callbacks 对话框，如图 6-83 所示。用户可以在左侧的回调函数列表中选择想要设置的回调，然后在右边的命令栏中输入相应的 MATLAB 命令。

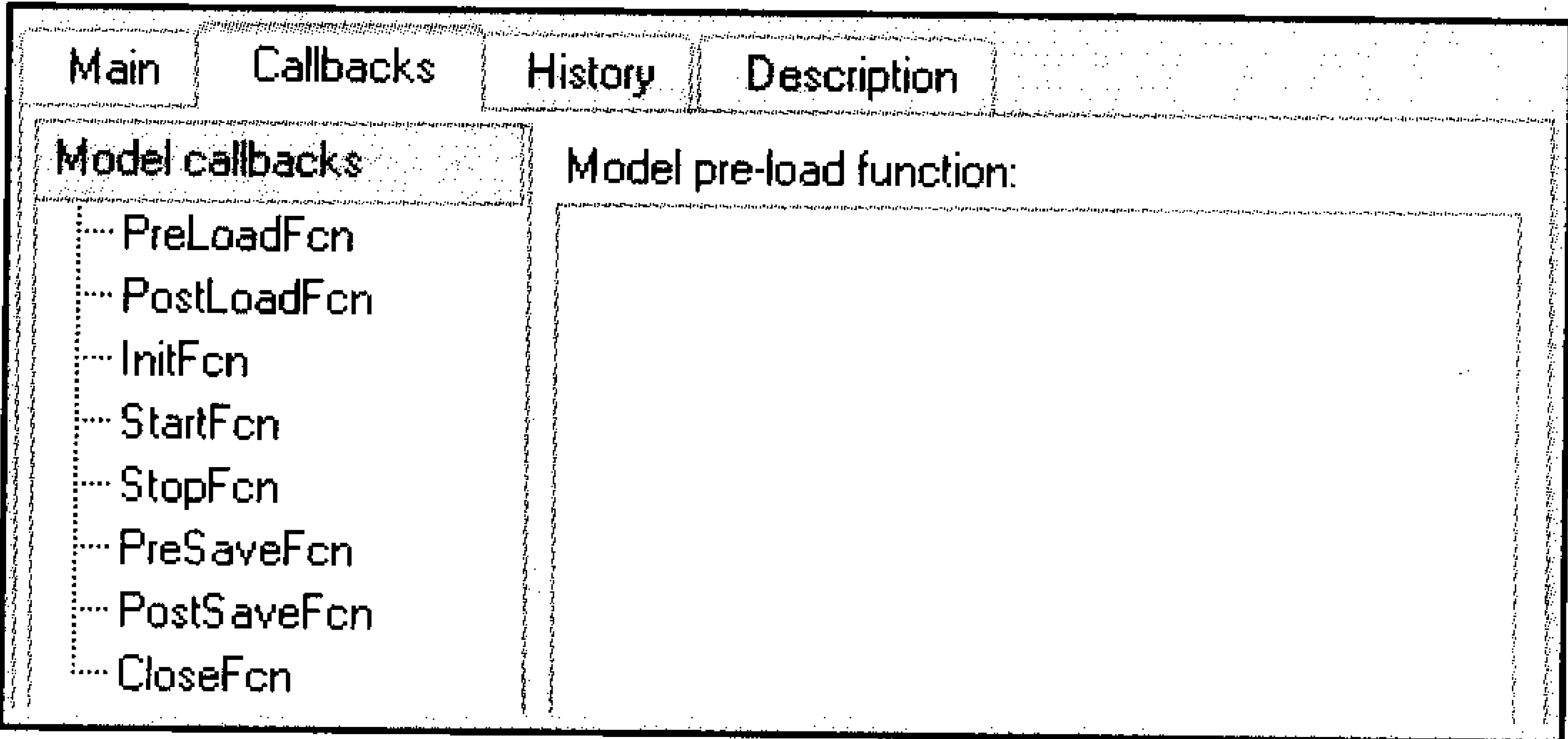


图 6-83　模型的 Callbacks 对话框

选中想要加载回调的模块，执行【Edit】→【Block properties】命令，打开模块属性对话框，单击“Callbacks”选项卡，则出现模块的 Callbacks 对话框，如图 6-84 所示。

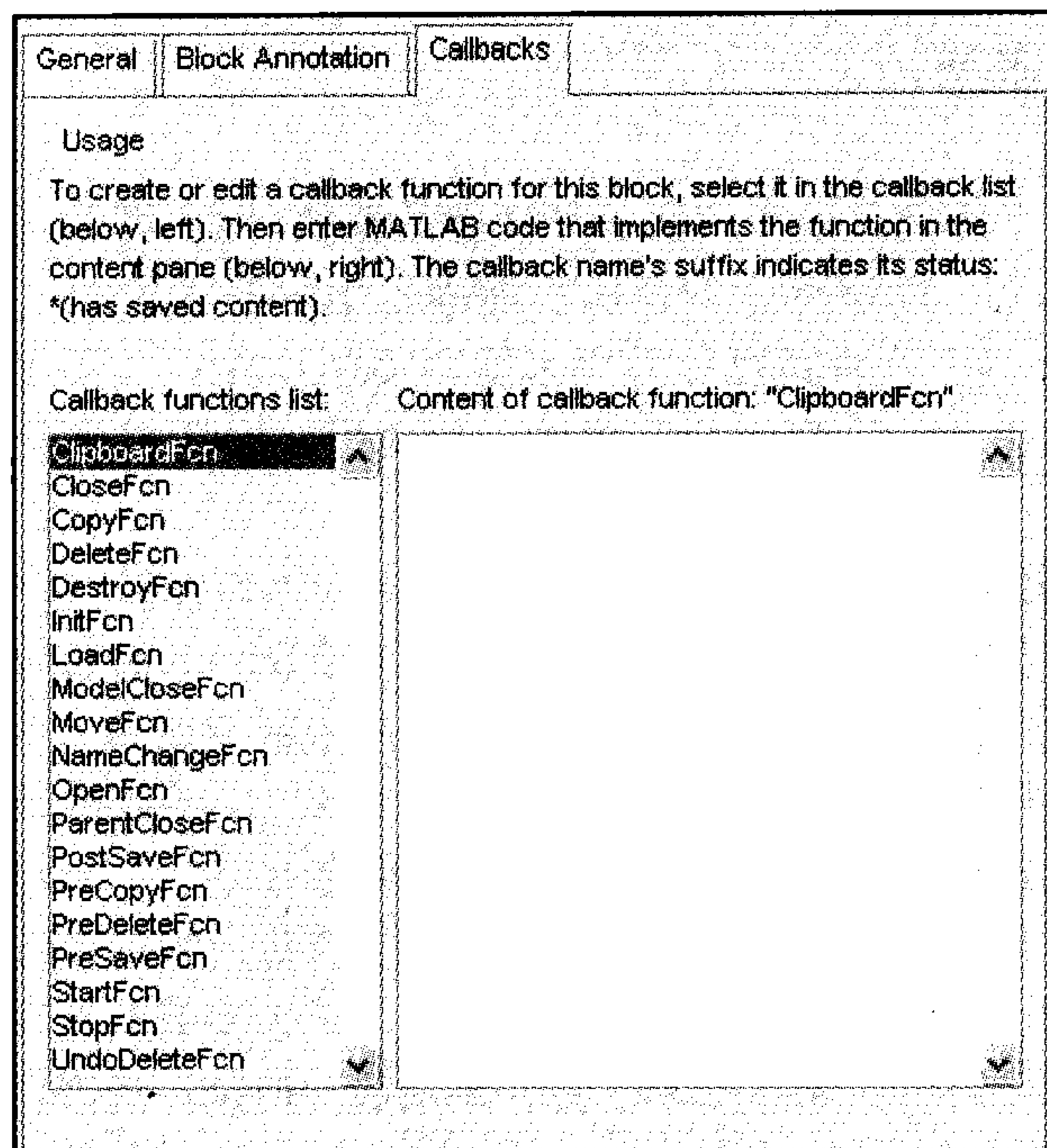


图 6-84　模块的 Callbacks 对话框

3. 回调举例

结合例 6.9 中 Simulink 模型 exa07_09，讲解模块回调函数的应用。针对模型 exa07_09 中的 Sine Wave 模块，使用如下回调加载命令：

```
>> set_param('exa07_09/Sine Wave','CloseFcn','Sine Wave 模块被关闭')
>> set_param('exa07_09/Sine Wave','CopyFcn','Sine Wave 模块被复制')
>> set_param('exa07_09/Sine Wave','DeleteFcn','Sine Wave 模块被删除')
>> set_param('exa07_09/Sine Wave','UndoDelete','撤销 Sine Wave 模块的删除操作')
```

读者也可以在 Sine Wave 模块的 Callbacks 对话框中输入相应的命令，如在回调函数 CloseFcn 的命令栏中输入“Sine Wave 模块被关闭”。

6.5　S-函数

S-函数无疑是 Simulink 最具魅力的地方，它完美地结合了 Simulink 框图简洁明快的特点和编程灵活方便的优点。它提供了增强和扩展 Simulink 能力的强大机制，同时也是使用 RTW（Real Time Workshop）实现实时仿真的关键。实际上，Simulink 许多模块所包含的算法均是用 S-函数写的，用户也可以编写自己的 S-函数，然后进行封装，得到具有特定功能的定制模块。S-函数支持 MATLAB、C、C++、FORTRAM 以及 Ada 等语言，使用这些语言，按照一定的规则就可以写出功能强大的模块。本节将介绍 S-函数的基本概念、工作原理，以及通过 M 文件、C MEX 文件和创建器编写和使用 S-函数的方法与实例。

6.5.1 S-函数基本概念

Simulink 模块库中包含有很多实现各种功能的模块，用户可以直接通过模块库调用它们，但是在很多情况下，库中自带的模块不能满足用户的需求，这时候用户希望可以自己设计模块以供使用。M 文件虽然能够用来编写 MATLAB 函数代码，但是它不具备与 Simulink 的接口，因此难以与 Simulink 模块一起使用。但是 S-函数可以使这种想法成为可能。

S-函数是一种描述 Simulink 模块的计算机语言，是系统函数（System Functions）的简称，S-函数通过一种特殊的调用规则，使得用户可以与 Simulink 解算器进行交互。这种交互和 Simulink 里自带模块与解算器的交互是一样的。运用 S-函数，用户可以把自己设计的模块与 Simulink 自带模块一起使用。从这个意义上来说，S-函数扩展了 Simulink 模块库的功能。而且 S-函数的形式很全面，可以用于不同性质的系统中，如连续、离散和混合系统。

用户可能会有如下的疑问：Simulink 已经提供了大量内置的系统模块，并且允许用户自定义模块，那么，为何还要使用 S-函数？诚然，对于大多数动态系统仿真分析而言，使用 Simulink 提供的模块即可实现，而无须使用 S-函数。但是，当需要开发一个新的通用的模块作为一个独立的功能单元时，使用 S-函数实现则是一种相当简便的方法。另外，由于 S-函数可以使用多种语言编写，因此可以将已有的代码结合进来，而不需要在 Simulink 中重新实现算法，从而在某种程度上实现了代码移植。此外，在 S-函数中使用文本方式输入公式、方程，非常适合复杂动态系统动态的数学描述，并且在仿真过程中可以对仿真进行更精确的控制。

简单来说，用户可以从如下几个角度来理解 S-函数：

- S-函数为 Simulink 的“系统”函数。
- 能够响应 Simulink 解算器命令的函数。
- 采用非图形化的方法实现一个动态系统。
- 可以开发新的 Simulink 模块。
- 可以与已有的代码相结合进行仿真。
- 采用文本方式输入复杂的系统方程。
- 扩展 Simulink 功能。M 文件 S-函数可以扩展图形能力，CMEX S-函数可以提供与操作系统的接口。
- S-函数的语法结构是为实现一个动态系统而设计的（默认用法），其他 S-函数的用法是默认用法的特例（如用于显示目的）。

S-函数的编写也相当灵活，可在 MATLAB 里用 M 文件、C 语言、C++语言、Ada 语言，或者 Fortran 语言编写。C 语言、C++语言、Ada 语言或者 Fortran 语言编写的 S-函数被编译成 MEX-文件，当需要的时候，它们被动态地连接到 MATLAB。总体说来，用 C 语言、C++语言、Ada 语言或者 Fortran 语言编写的 S-函数比 MATLAB 语言编写的 S-函数具有更强的控制能力。

在编写 S-函数过程中经常会遇到以下几个基本概念，很好地理解它们对于编写 S-函数会有很大的帮助。这些概念是仿真例程（Routines）、直接反馈（Direct Feedthrough）、采样

时间和偏移量（Setting Sample Times and Offsets），以及动态可变维数信号输入（Dynamically Sized Inputs），它们是编写 S-函数的基础。

1．仿真例程

Simulink 在仿真的特定阶段调用对应的 S-函数功能模块（函数）来完成不同的任务，如初始化、计算输出、更新离散状态、计算导数、结束仿真等，这些功能模块（函数）称为仿真例程或者回调函数（Call Backfunctions）。表 6-18 所示列出了 S-函数例程函数和对应的仿真阶段。

表 6-18 S-函数例程及其仿真阶段

S-函数仿真例程名称	仿 真 阶 段	S-函数仿真例程名称	仿 真 阶 段
mdlInitialization	初始化	mdlUpdate	更新离散状态
mdlGetTimeofNextVarHit	计算下一个采样点	mdlDerivatives	计算导数
mdlOutput	计算输出	mdlTerminate	结束仿真

2．直接反馈

直接反馈就是输出（或者可变采样时间模块的可变采样时间）受输入信号的直接控制。当存在如下的两种情况时，系统需要直接反馈。

- 某一时刻的系统输出中包含某一时刻的系统输入。
- 系统是一个变采样时间系统（Variable Sample System），且采样时间计算与输入相关。

正确设置反馈标志是非常重要的，因为这不仅关系到系统模型中系统模块的执行顺序，还关系到对代数环的检测与处理。

3．采样时间和偏移量

在离散时间系统内采样时间 time，采样时间间隔 sample_time_value 和偏移量 offset_time 之间的关系如下：

time＝(*n*×sample_time_value)+offset_time

其中，*n* 表示第 *n* 个采样点。

Simulink 在每一个采样点上调用 mdlOutput 和 mdlUpdate 例程。系统采样时间还可以继承自驱动模块、目标模块或者系统最小采样时间，这种情况下采样时间值应该设置为–1，或者 INHERITED_SAMPLE_TIME。

4．动态可变维数信号输入

S-函数支持动态可变维数信号的输入。S-函数输入变量的维数决定于驱动 S-函数模块的输入信号的维数。所以当仿真开始的时候，需要先估计 S-函数输入信号的维数。在 M 文件 S-函数中动态设置输入信号维数时，应该把 sizes 数据结构的对应成员设置为–1 或 DYNAMICLLY_SIZED。在 C 文件 S-函数中需要调用函数 ssSetInputPortWidth 来动态设置输入信号维数。其他的如状态维数和输出维数同样是动态可变的。

6.5.2 S-函数工作原理

了解S-函数的工作原理对于用户编写S-函数无疑是非常有帮助的，对于理解Simulink的整个仿真原理也是很有益的。这里先简单地介绍作为S-函数的数学基础——状态方程，然后介绍S-函数的仿真流程。

1．S-函数的数学模型描述

无论是线性系统还是非线性系统，Simulink对其模型的描述可以简化为如图6-85所示的图形。

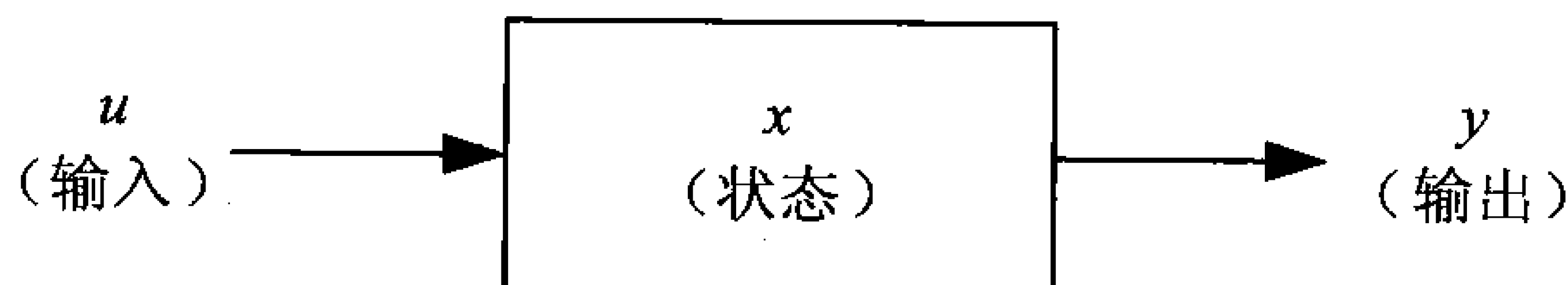


图6-85 Simulink系统模型的简化模块

图6-85中的输入向量u、输出向量y和状态量x可以用下述状态方程来描述：

状态方程：$\overline{x} = f_c(x,u,t)$

输出方程：$y = g(x,u,t)$

状态方程描述了状态变量的一阶导数与状态变量、输入量之间的关系。n阶系统具有n个独立的状态变量，系统状态方程则是n个联立的一阶微分方程或者差分方程。对于一个系统，由于所选择的状态变量不同，会导出不同的状态方程，因此状态方程的形式不是唯一的。输出方程描述了输出与状态变量、输入量之间的关系。输出量根据任务的需要确定。一个典型的线性系统的状态方程可以用矩阵的形式描述为：

状态方程：$\overline{x} = Ax + Bu$

输出方程：$y = Cx + Du$

其中，A、B、C、D分别是状态矩阵、输入矩阵、输出矩阵和前馈矩阵。

S-函数同样是一个Simulink模块。它的以下几个例程函数清楚地体现了状态空间所描述的特性。

- 连续状态方程描述的S-函数例程mdlDerivatives。状态向量的一阶导数是状态x、输入u和时间t的函数。在S-函数中，使用mdlDerivatives例程来计算状态x的一阶导数，并将结果返回解算器，供其积分使用。
- 离散状态方程描述的S-函数例程mdlupdale。下一步状态x_{k+1}的值依赖于当前状态x_k、输入u和时间t等3个变量。这时使用mdlupdale例程来计算x_{k+1}，并将结果返回解算器。
- 输出方程描述的S-函数例程mdloutputs。在输出方程不包含任何动态方程（微分或差分）在内，其输出y是状态x、输入u和时间t的函数。输出值是在mdloutputs中计算的，并通过解算器传递给其他模块。

2．S-函数的仿真流程

S-函数是Simulink的重要组成部分，它的仿真过程包含在Simulink仿真过程之中。当

然，S-函数的仿真流程也包括初始化和运行两个阶段。

初始化工作完成以后，在每一个仿真步长（Time Step）内完成一次求解，如此反复，形成一个仿真循环，直到仿真结束。为了提高精度，解算器一方面会检查两次仿真结果一致性。当误差超过解算器所设置的容忍误差（Tolerance）时，会减小仿真步长，重新进行仿真计算。另一方面，解算器还会检查是否发生过零事件。一旦检测到过零事件，Simulink 在发生过零事件的变量或者状态的当前值和前一个值之间进行插值运算。

S-函数的仿真流程如下：

（1）初始化。在仿真开始前，Simulink 将对 S-函数进行初始化处理，具体内容包括：初始化结构体 Simstruct（它包含了 S-函数的所有信息），设置输入输出端口数，设置采样时间和分配存储空间。在这一步中可使用的例程有 mdlInitializeConditions、mdlInitializeSize 和 mdlInitializeSampleTime。

（2）计算下一个采样时间点。当使用变步长解算器进行仿真时，则需要使用 mdlGetTimeofNextVarHit 例程计算下一个采样时间点，即计算下一步的仿真步长。

（3）计算输出。利用 mdlOutputs 例程计算所有输出端口的输出值。

（4）更新状态。在每个仿真步长中，都需要执行一次更新状态例程 mdlUpdate，因此可以将需要更新的内容添加到 mdlUpdate 中，例如，离散状态的更新。

（5）数值积分。在连续状态的求解和非采样过零点时使用。如果 S-函数存在连续状态，Simulink 就在 minor step time 内调用 mdlDerivatives 和 mdlOutput 两个例程。如果存在非采样过零点，Simulink 将调用 mdlOutput 和 mdlZeroCrossings 例程以检测过零点。

（6）仿真结束。S-函数使用 mdlTerminate 例程来处理仿真结束时的工作。

从 S-函数的仿真流程可以看出，Simulink 在每个仿真阶段都会调用相应的例程。

6.5.3 用 M 文件编写 S-函数

S-函数是由一些仿真功能模块（例程）组成的。它可以是 M 文件，也可以是 MEX 文件。M 文件 S-函数结构明晰，易于理解，书写方便，且可以调用丰富的 MALTAB 函数，对于一般的应用，使用 M 文件编写 S-函数就足够了。

1．M 文件 S-函数的工作流程

M 文件 S-函数的工作流程与上一节介绍的 S-函数的仿真流程一致。它通过标志 Flag 来调用例程函数，控制仿真流程。当 Flag=0 时，程序初始化；当 Flag=1 时，计算系统状态变量的导数；当 Flag=2 时，更新离散系统状态；当 Flag=3 时，计算系统的输出；当 Flag=4 时，请求 S-函数提供下一步的采样时间（这个例程在单采样速率系统中不被调用）；当 Flag=9 时，处理仿真结束时的工作。

2．M 文件 S-函数模板

Simulink 为我们编写 S-函数提供了各种模板文件，其中定义了 S-函数完整的框架结构，用户可以根据自己的需要加以剪裁。在编写 M 文件 S-函数时，推荐使用 S-函数模板文件 sfuntmpl.m。这个文件包含了一个完整的 M 文件 S-函数，它由 1 个主函数和 6 个子函数构成。通过 Switch-Case 条件执行语句结构，主函数根据标志变量 Flag 的数值将执行

流程转移到相应的子函数，即例程函数。Flag 标志量作为主函数的参数由 Simulink 引擎调用时给出。

sfuntmpl.m 位于$\MATLAB\R2007a\toolbox\Simulink\blocks 目录下。要打开该文件，用户可以在相应目录下找到该文件并双击它，或者在 MATLAB 命令窗口中输入：

```
>> edit sfuntmpl
```

为了方便读者理解，在此给出该模板文件基本框架，并配置相应的中文说明：

```
% 主函数
% 主函数包含 4 个输出：sys 数组包含某个子函数返回的值，它的含义随着调用
% 子函数的不同而不同：x0 为所有状态的初始化向量；str 是保留参数，总是一
% 个空矩阵；Ts 返回系统采样时间，函数的 4 个输入分别为采样时间 t，状态 x，
% 输入 u 和仿真流程控制标志变量 flag。

% sfuntmpl 是 M-文件 S-函数模板
function [sys,x0,str,ts]=sfuntmpl(t,x,u,flag)
% 标志 flag 控制仿真流程
switch flag,
  % 初始化，调用“模块初始化”子函数
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
  % 连续状态变量计算，调用“计算模块导数”子函数
  case 1,
    sys=mdlDerivatives(t,x,u);
  % 更新，调用“更新模块离散状态”子函数
  case 2,
    sys=mdlUpdate(t,x,u);
  % 输出，调用“计算模块输出”子函数
  case 3,
    sys=mdlOutputs(t,x,u);
  % 计算下一时刻采样点，调用“计算下一个采样时间点”子函数
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
  % 结束，调用“结束仿真”子函数
  case 9,
    sys=mdlTerminate(t,x,u);
  % 其他 flags 数值时的处理方法
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
% 主函数结束

%下面是各个子函数，即各个仿真例程
%1、初始化例程子函数：提供状态、输入、输出、采样时间数目和初始状态的值。
% M 文件 S-函数必须有该子函数。初始化阶段，标志变量首先被置为 0，S-函数
% 第一次执行时，mdlInitializes 子函数首先被调用。这个子函数为系统提供下列
%S-函数信息：
% 连续状态的个数
% 离散状态的个数
% 输出的个数
% 输入的个数
```

```
% 是否直接反馈：当采用直接反馈时，数值为 1，否则为 0
% 采样时间的个数；每个系统至少有一个采样时间
% 上述的 S-函数的信息是通过数据结构 size 来表示的
% 在该函数中用户还应该提供初始状态 x0，采样时间 ts。ts 是一个 m×2 的矩阵，
% 其中第 k 行包含了对应与第 k 个采样时间的采行周期值和偏移量。另外，在该
% 子函数中 str 设置为空，str 是保留变量，暂时没有任何意义。
Function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;      % 生成 sizes 数据结构
sizes.NumContStates   =0,      % 连续状态数，默认为 0
sizes.NumDiscStates   =0;      % 离散状态数，默认为 0
sizes.NumOutputs     =0;      % 输出量个数，默认为 0
sizes.NumInputs    =0;      % 输入量个数，默认为 0
sizes.DirFeedthrough=1;      % 是否存在直接馈通。1：存在；0：不存在，默认为 1
sizes.NumSampleTimes=1;      % 采样时间个数，至少是一个
sys=simsizes(sizes);      % 返回 sizes 数据结构所包含的信息
x0=[];                    % 设置初始状态
str=[];                   % 保留变量置空
ts=[0 0];             % 采样时间：[采样周期 偏移量]，采样周期为 0 表示是连续系统

%2、计算导数例程子函数：给定 t，x，u，计算连续状态的导数。该子函数用于计
% 算连续系统的状态方程，可以不存在。
Function sys=mdlDerivatives(t,x,u)
Sys=[];      % 表示系统状态导数，即 dx

%3、状态更新例程子函数：给定 t，x，u 计算离散状态的更新。每一步仿真
% 必然调用该子函数，不论是否有意义。它除了用于描述离散系统的状态方
% 程之外，还可以在子函数中填入其他每个仿真步长都需要执行的代码。
function sys=mdlUpdate(t,x,u)
sys=[];      % sys 表示下一个离散状态，即 x(k+1)

%4、计算输出例程子函数：给定 t，x，u，计算输出。该子函数必须存在，
% 用户可以在此描述系统的输出方程
function sys=mdlOutputs(t,x,u)
sys=[];     % sys 表示系统输出，即 y

%5、计算下一个采样时间，仅在系统是变采样时间系统时调用
Function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime=1;               % 设置下一次的采样时间是 1s 以后
sys=t+sampleTime;           % sys 表示下一个采样时间点

%6、仿真结束时要调用的例程函数，在仿真结束时，用户可以在此完成
% 仿真结束时的工作
function sys=mdlTerminate(t,x,u)
sys=[];             % sys 表示仿真结束
```

下面我们通过连续系统、离散系统和混合系统，讲述使用 M 文件 S-函数实现这些系统的方法。

3．连续系统的 S-函数描述

用 S-函数实现一个连续系统时，首先 mdlInitilizeSizes 子函数应当做适当的修改，包括

确定连续状态的个数、状态初始值和采样时间设置。另外，还需要编写 mdlDerivatives 子函数，将状态的导数向量通过 sys 变量返回。如果系统状态不止一个，可以通过索引 x(1)，x(2)得到各个状态。当然，对于多个状态，就会有多个导数与之对应。在这种情况下，sys 为一个向量，其中包含了所有连续状态的导数。与前文所述一样，修改后的 mdlOutputs 中应包含系统的输出方程。下面使用 S-函数实现一个最简单的连续系统模块——积分器。

例 6.10 用 M 文件编写 S-函数代替连续系统态方程：

$$\begin{cases} \overline{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

其中，$A = \begin{bmatrix} -0.09 & -0.01 \\ 1 & 0 \end{bmatrix}$，$B = \begin{bmatrix} 1 & -7 \\ 0 & -2 \end{bmatrix}$，$C = \begin{bmatrix} 0 & 2 \\ 1 & 5 \end{bmatrix}$，$D = \begin{bmatrix} 3 & 0 \\ 1 & 0 \end{bmatrix}$。

实现该连续系统的步骤如下。

（1）借助 sfuntmpl.m 模板函数，编写该连续系统的 S-函数如下：

```
function [sys,x0,str,ts] = exa07_10_msf(t,x,u,flag)
% 利用 M 文件编写该连续系统的 S-函数
%        x' = Ax + Bu
%        y  = Cx + Du

% 产生一个连续线性系统：
A=[-0.09     -0.01
     1              0];
B=[ 1     -7
     0     -2];
C=[ 0      2
     1     -5];
D=[-3      0
     1      0];

switch flag,
  % 初始化
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D);

   % 连续系统状态求导
  case 1,
    sys=mdlDerivatives(t,x,u,A,B,C,D);

  % 系统输出
  case 3,
    sys=mdlOutputs(t,x,u,A,B,C,D);

  % flags=2，4，9 时的处理方法
  case { 2, 4, 9 },
    sys = [];

  % 当 flags 为其他数值时的处理方法
  otherwise
```

```
    error(['Unhandled flag = ',num2str(flag)]);

end
% 主程序结束

% mdlInitializeSizes
% 返回 S-函数的 sizes, initial conditions 和 sample times 信息
function [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D)

sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
x0  = zeros(2,1);
str = [];
ts  = [0 0];
% 结束 mdlInitializeSizes

% mdlDerivatives
% 返回连续系统状态的导数
function sys=mdlDerivatives(t,x,u,A,B,C,D)

sys = A*x + B*u;
% end mdlDerivatives

% mdlOutputs
% 返回模块的输出
function sys=mdlOutputs(t,x,u,A,B,C,D)

sys = C*x + D*u;

% 结束 mdlOutputs
```

（2）建立连续系统 Simulink 模型框架，如图 6-86 所示，保存为 exa07_10.mdl，同时将 exa07_10_msf.m S-函数文件也保存在 exa07_10.mdl 相同的目录下，并将该目录设置为 MATLAB 当前工作目录。

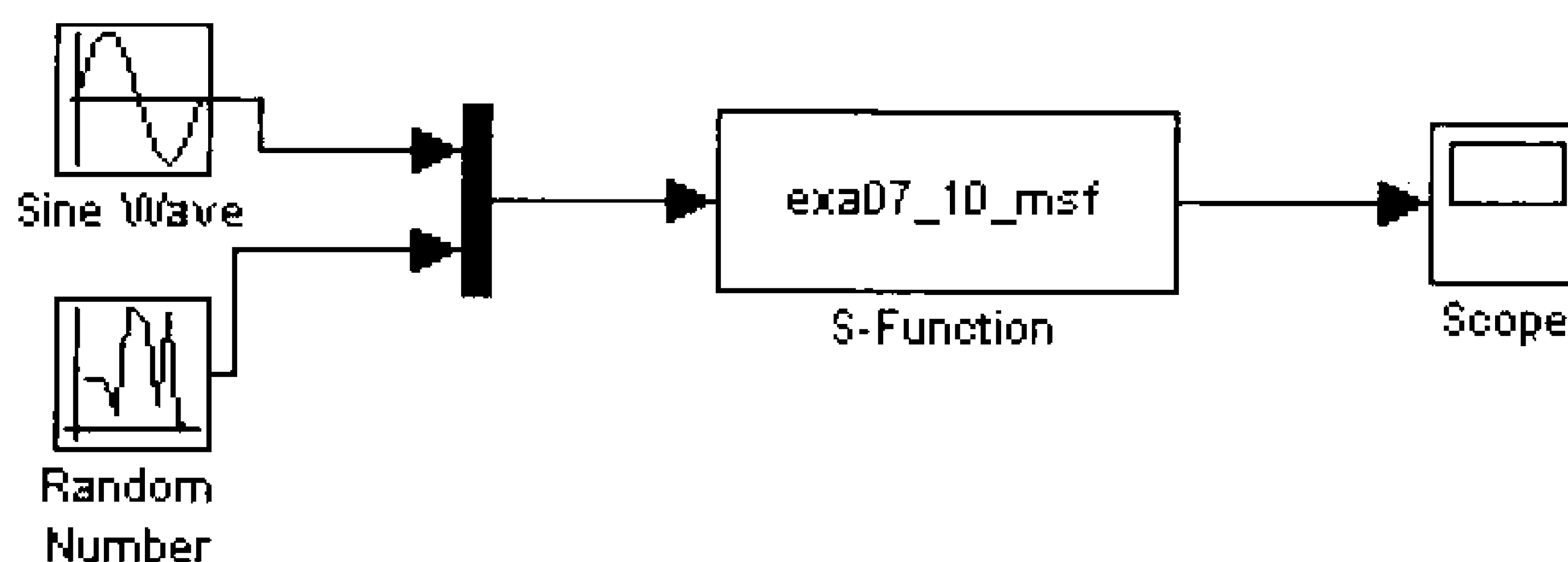


图 6-86 系统模型框架

（3）模块 Sine Wave 和模块 Random Number 都采用默认设置，S-Functions 模块在其参数设置的 S-Functions name 栏设置为 exa07_10_msf。

（4）运行 exa07_10.mdl 模型，结果如图 6-87 所示。

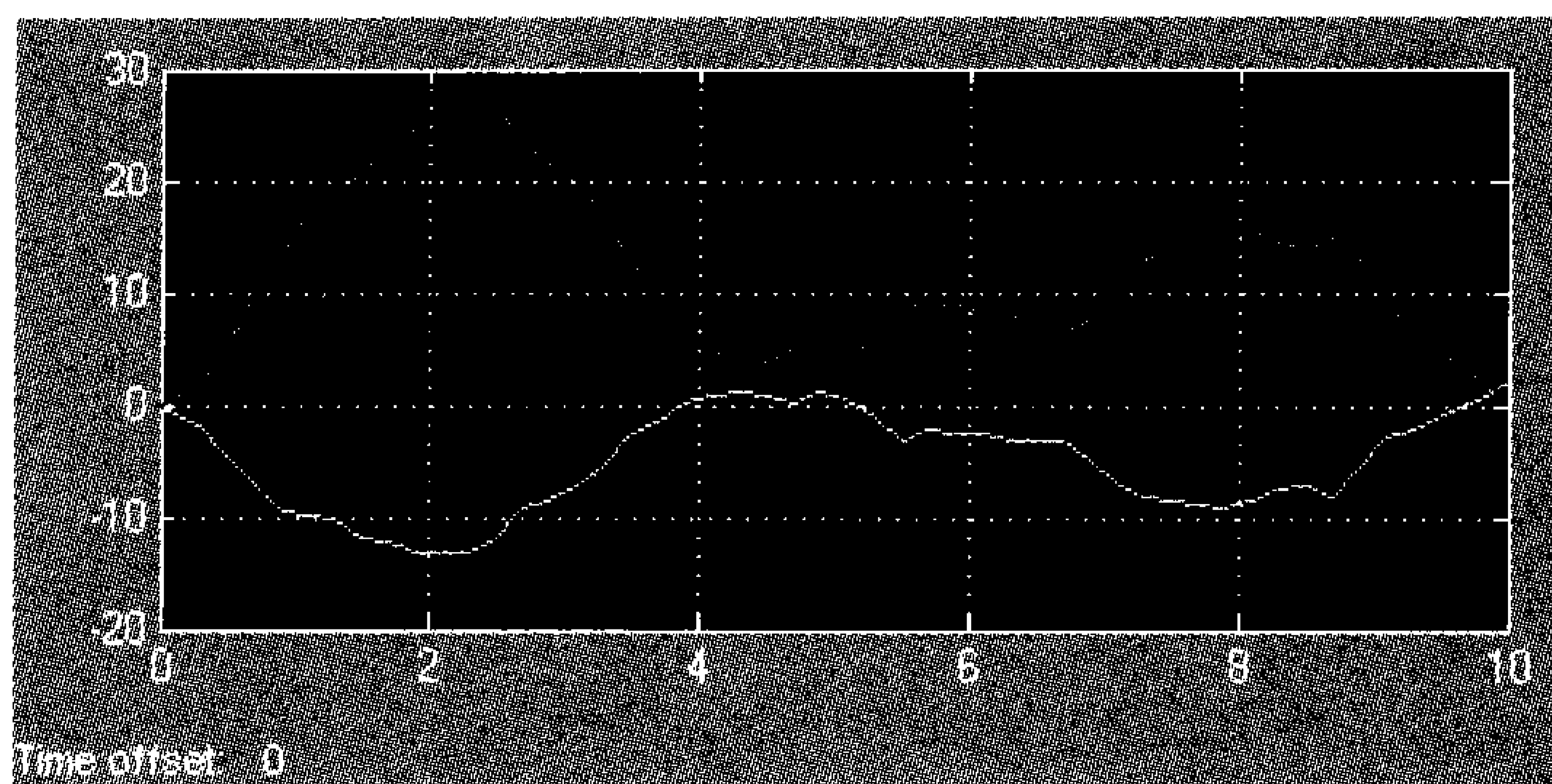


图 6-87　M 文件 S-函数建立的连续系统模型输出结果

4．离散系统的 S-函数描述

用 S-函数模板实现一个离散系统时，首先对 mdlInitializeSizes 子函数进行修改，声明离散状态的个数，对状态进行初始化，确定采样时间等。然后再对 mdlUpdate 和 mdloutputs 函数做适当的修改，分别输入要表示的系统的离散状态方程和输出方程即可。

例 6.11 用 M 文件编写 S-函数代替离散系统态方程：

$$\begin{cases} x(n+1) = Ax(n) + Bu(n) \\ y(n) = Cx(n) + Du(n) \end{cases}$$

其中，$A=\begin{bmatrix} -1.3839 & -0.5097 \\ 1 & 0 \end{bmatrix}$，$B=\begin{bmatrix} -2.559 & 0 \\ 0 & 4.2382 \end{bmatrix}$，$C=\begin{bmatrix} 0 & 2.0761 \\ 0 & 7.7891 \end{bmatrix}$，$D=\begin{bmatrix} -0.8141 & -2.9334 \\ 1.2426 & 0 \end{bmatrix}$。

实现该离散系统的步骤如下。

（1）借助 sfuntmpl.m 模板函数，编写该离散系统的 S-函数如下：

```
function [sys,x0,str,ts] = exa07_11_msf(t,x,u,flag)
% 利用 M 文件编写该离散系统的 S-函数
%        x(n+1) = Ax(n) + Bu(n)
%        y(n)    = Cx(n) + Du(n)

% 产生一个离散线性系统：
A=[-1.3839    -0.5097
     1.0000          0];
B=[-2.5559           0
           0    4.2382];
C=[        0    2.0761
           0    7.7891];
D=[   -0.8141    -2.9334
       1.2426          0];
```

```
switch flag,

  % 初始化
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes(A,B,C,D);

  % 模块更新
  case 2,
    sys = mdlUpdate(t,x,u,A,B,C,D);

  % 模块输出
  case 3,
    sys = mdlOutputs(t,x,u,A,C,D);

  % 仿真结束
  case 9,
    sys = []; % do nothing

  % 当 flags 为其他数值时的处理方法
  otherwise
    error(['unhandled flag = ',num2str(flag)]);
end
% 主函数结束

% mdlInitializeSizes
% 返回 S-函数的 sizes，initial conditions 和 sample time 信息
function [sys,x0,str,ts] = mdlInitializeSizes(A,B,C,D)
sizes = simsizes;
sizes.NumContStates    = 0;
sizes.NumDiscStates    = size(A,1);
sizes.NumOutputs       = size(D,1);
sizes.NumInputs        = size(D,2);
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0    = ones(sizes.NumDiscStates,1);
str = [];
ts    = [1 0];
% 结束 mdlInitializeSizes

% mdlUpdate
% 更新离散状态，sample time hits 和 major time step 参数
function sys = mdlUpdate(t,x,u,A,B,C,D)
sys = A*x+B*u;
%结束 mdlUpdate

% mdlOutputs
% 返回 S-函数的输出变量
function sys = mdlOutputs(t,x,u,A,C,D)
```

```
sys = C*x+D*u;
%结束 mdlOutputs
```

（2）建立的离散系统 Simulink 模型框架如图 6-86 所示，除 S-Functions 模块的 S-函数模块的名称设置为 exa07_11_msf 之外，其他参数设置与例 6.10 的 Simulink 模型相同。

（3）运行 exa07_11.mdl 模型，结果如图 6-88 所示。

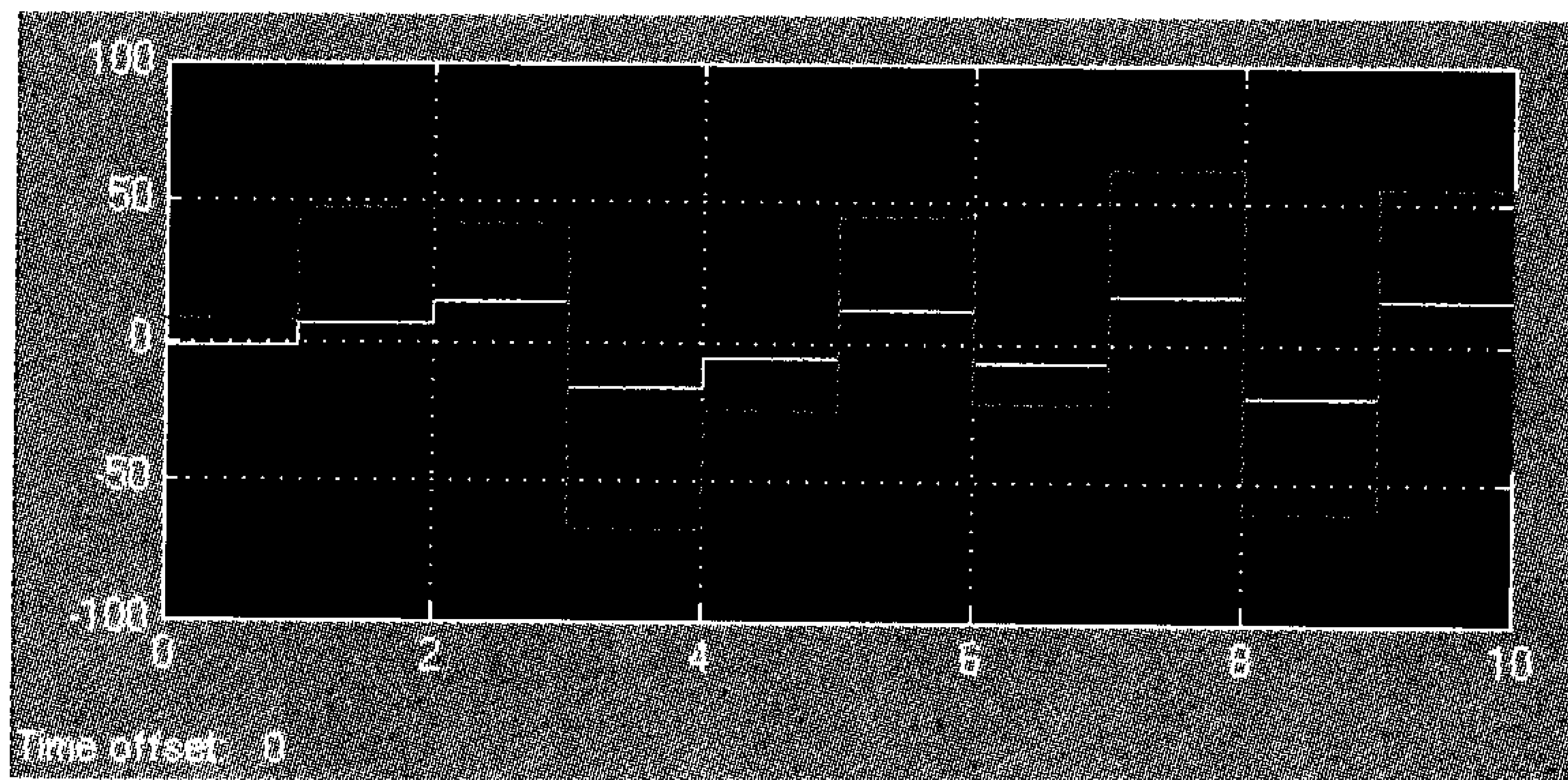


图 6-88　M 文件 S-函数建立的离散系统模型输出结果

5．混合系统的 S-函数描述

所谓混合系统，就是既包含离散状态，又包含连续状态的系统。Simulink 根据 flag 的具体数值判断系统是计算连续部分还是离散部分，并调用相应的子函数，Simulink 在处理混合系统时将同时调用 S-函数的 mdlUpdate、mdlOutput 和 mdlGetTimeOfNextVarHit 子函数。对于离散系统而言，在 mdlUpdate、mdloutput 中需要判断是否需要更新离散状态和输出。因为对于离散状态并不是在所有的采样点上都需要更新，否则就是一个连续系统了。

例 6.12 利用 M 文件 S-函数要实现混合系统的模型如图 6-89 所示。

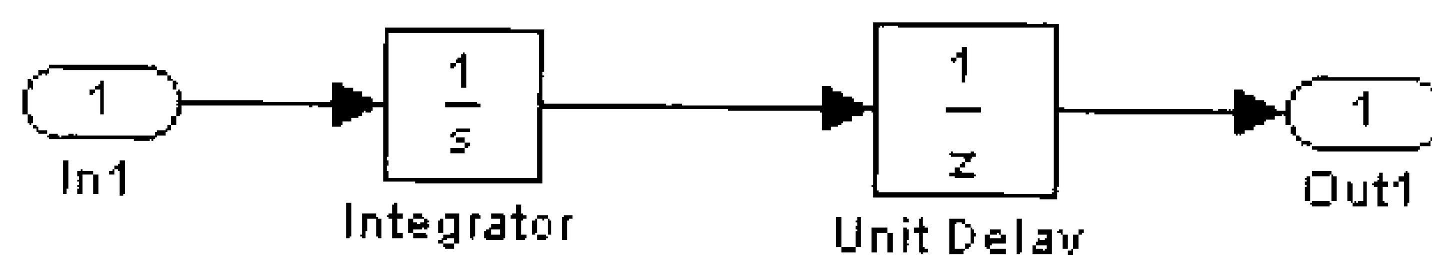


图 6-89　利用 M 文件 S-函数要实现混合系统模型

实现该混合系统的步骤如下。

（1）借助 sfuntmpl.m 模板函数，编写该离散系统的 S-函数如下：

```
function [sys,x0,str,ts] = exa07_12_msf(t,x,u,flag)
%MIXEDM An example integrator followed by unit delay M-file S-Function
% 利用 M 模板文件编写 S-函数，实现单位延迟（1/z）积分（1/s）的连续离散混合系统

% 设置单位延迟的采样周期和偏移量
dperiod = 1;
doffset = 0;

switch flag
  % 初始化
  case 0
```

```
        [sys,x0,str,ts]=mdlInitializeSizes(dperiod,doffset);

    % 连续系统状态求导
    case 1
        sys=mdlDerivatives(t,x,u);

    % 系统模型更新
    case 2,
        sys=mdlUpdate(t,x,u,dperiod,doffset);

    % 系统输出
    case 3
        sys=mdlOutputs(t,x,u,doffset,dperiod);

    % 仿真结束
    case 9
        sys = [];           % 什么也不做

    otherwise
        error(['unhandled flag = ',num2str(flag)]);

end
% 主程序结束

% mdlInitializeSizes
% 返回 S-函数的 sizes，initial conditions 和 sample times 信息
function [sys,x0,str,ts]=mdlInitializeSizes(dperiod,doffset)

sizes = simsizes;
sizes.NumContStates    = 1;          % 一个连续状态
sizes.NumDiscStates    = 1;          % 一个离散状态
sizes.NumOutputs       = 1;          % 一个输出
sizes.NumInputs        = 1;          % 一个输入
sizes.DirFeedthrough = 0;            % 没有直接反馈
sizes.NumSampleTimes = 2;            % 两个采样时间
sys = simsizes(sizes);
x0   = ones(2,1);
str = [];
ts   = [0 0;               % 一个采样时间是[0 0]表示是连续系统
                           % 离散系统的采样时间就是主程序开始所设置的两个变量
       dperiod doffset];

% 结束 mdlInitializeSizes

% mdlDerivatives
% 计算连续系统状态变量导数
function sys=mdlDerivatives(t,x,u)
sys = u;
% 结束 mdlDerivatives
```

```
% mdlUpdate
% 更新离散状态，sample time hits 和 major time step 参数
function sys=mdlUpdate(t,x,u,dperiod,doffset)
% 积分器输出的下一个离散系统状态
if abs(round((t - doffset)/dperiod) - (t - doffset)/dperiod) < 1e-8
   sys = x(1);
else
   sys = [];
end
% 结束 mdlUpdate

% mdlOutputs
% 返回 S-函数的输出向量
function sys=mdlOutputs(t,x,u,doffset,dperiod)
if abs(round((t - doffset)/dperiod) - (t - doffset)/dperiod) < 1e-8
   sys = x(2);
else
   sys = [];
end
% 结束 mdlOutputs
```

（2）建立的混合系统 Simulink 模型框架如图 6-90 所示，除 S-函数模块的名称设置为 exa07_12_msf 外，其他参数设置与例 6.10 的 Simulink 模型相同。

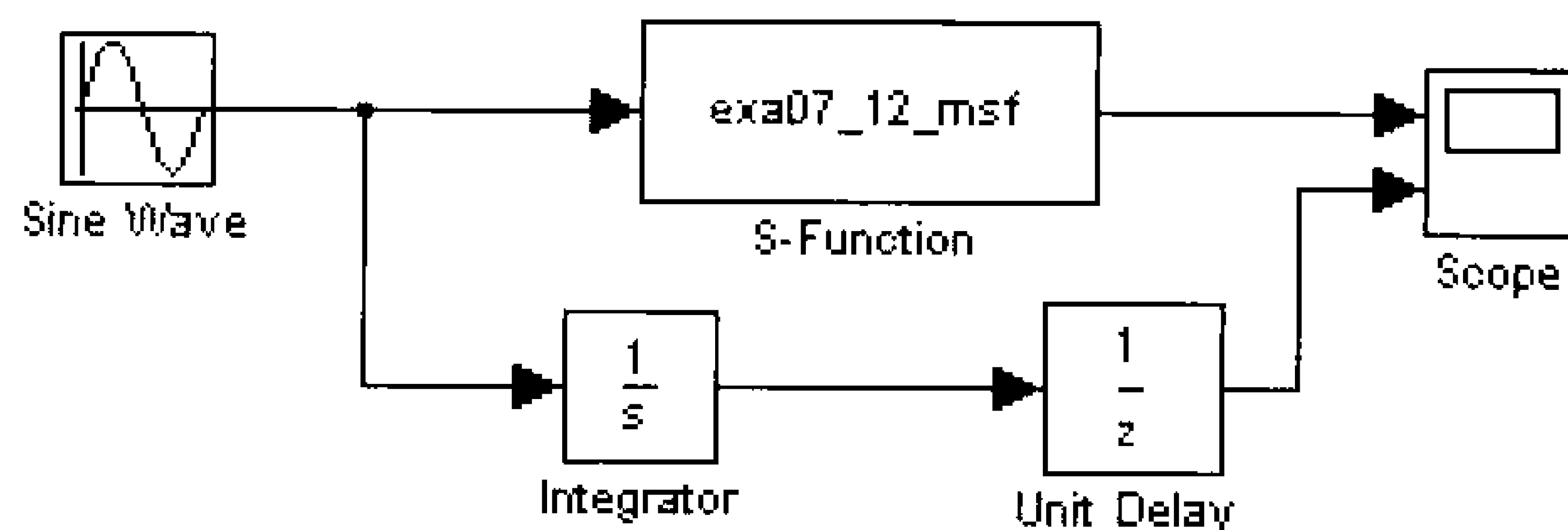


图 6-90　混合系统的 Simulink 模型框架

（3）运行 exa07_12.mdl 模型，结果如图 6-91 所示。

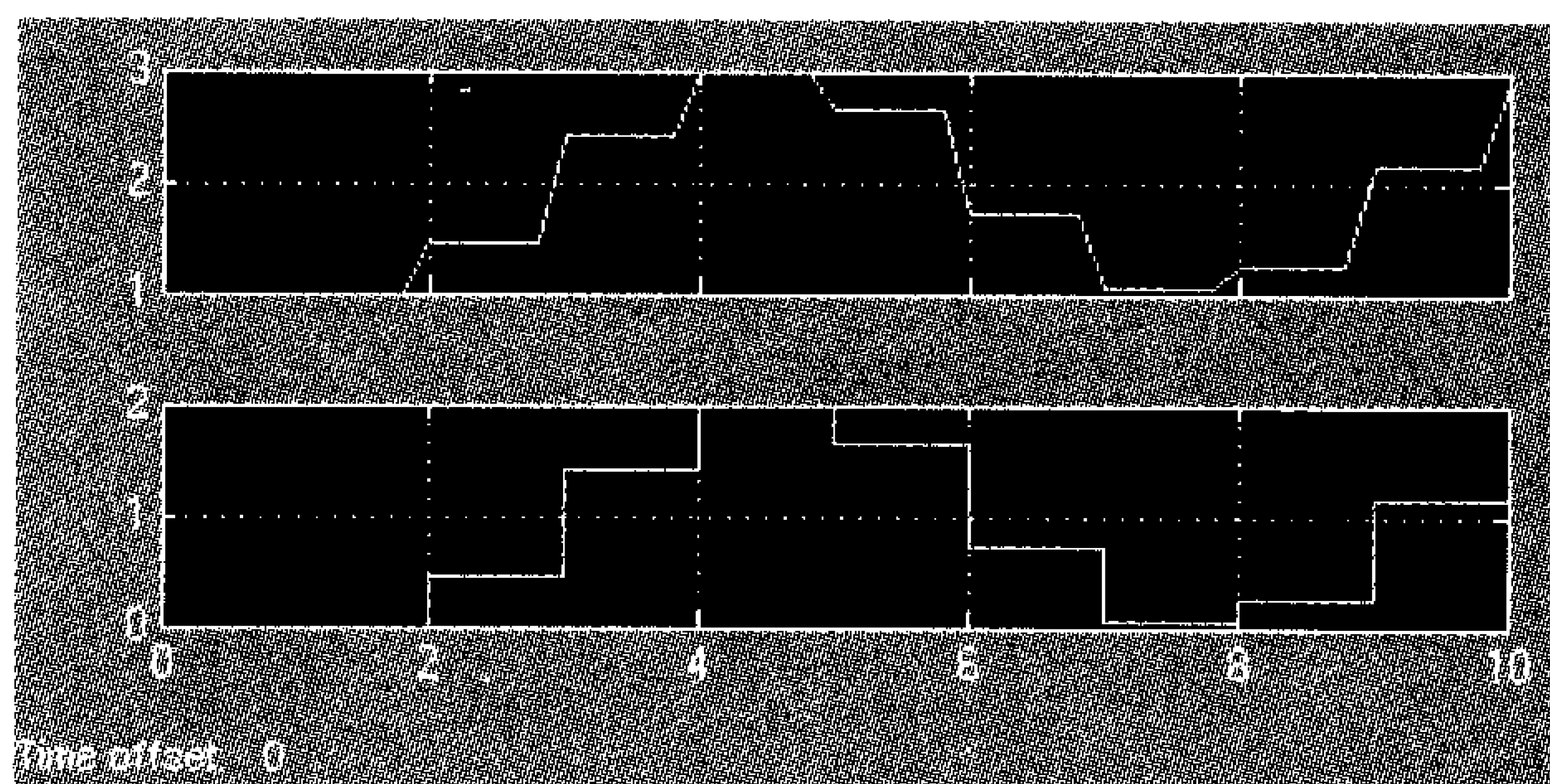


图 6-91　M 文件 S-函数建立的混合系统模型输出结果

6.5.4 用 C MEX 文件编写 S-函数

C MEX 文件的 S-函数与 M 文件的 S-函数在函数结构和子函数方面基本相同。但是 C 语言的灵活性使得 C MEX 能够实现比 M 文件在算法上更强的功能。Simulink 同样为 C MEX 文件的编写提供了模板 sfuntmpl_basic.c，该文件位于$\MATLAB\R2007a\Simulink\src 目录下，可以用 MATLAB 文本编辑器打开。

为了方便读者理解，在此给出该模板文件基本框架，并配置相应的中文说明。

```
/* 2 级 S-函数的 C MEX 模板文件 sfuntmpl_basic.c */
#define S_FUNCTION_NAME   sfuntmpl_basic
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"

/* S-Function 方法 */

/* 函数 mdlInitializeSizes
 * Simulink 利用该信息决定 S-函数模块的特性（如输入、输出和状态等变量数目等）
*/
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);  /* 需要参数的数目 */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        /* 当需要参数与实际参数数目相等时，则返回 */
        return;
    }

    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);

    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 1);
    ssSetInputPortRequiredContiguous(S, 0, true);  /* 指定信号的入口 */
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 1);

    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, 0);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);

    ssSetOptions(S, 0);
}

/* 函数 mdlInitializeSampleTimes
 * 指定 S-函数的采样时间 */
```

```
static void mdlInitializeSampleTimes(SimStruct *S)
{
     ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
     ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS    /* 改变到#undef 以移除函数 */
#if defined(MDL_INITIALIZE_CONDITIONS)
/* mdlInitializeConditions
 * 初始化 S-函数模块的连续或离散状态   */
  static void mdlInitializeConditions(SimStruct *S)
  {
  }
#endif /* MDL_INITIALIZE_CONDITIONS */

#define MDL_START   /* 改变到#undef 以移除函数 */
#if defined(MDL_START)
  /*函数 mdlStart
   * 该函数在模型开始运行时执行 */
  static void mdlStart(SimStruct *S)
  {
  }
#endif /*   MDL_START */

/* 函数 mdlOutputs
 *计算 S-函数模块的输出 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    const real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
    real_T          *y = ssGetOutputPortSignal(S,0);
    y[0] = u[0];
}

#define MDL_UPDATE  /* 改变到#undef 以移除函数 */
#if defined(MDL_UPDATE)
  /* 函数 mdlUpdate
   * 该函数在每一步主要积分时间段被调用，以更新离散系统的状态 */
  static void mdlUpdate(SimStruct *S, int_T tid)
  {
  }
#endif /* MDL_UPDATE */



#define MDL_DERIVATIVES     /* 改变到#undef 以移除函数 */
#if defined(MDL_DERIVATIVES)
  /* 函数 mdlDerivatives
   * 计算连续模型系统状态的导数 */
  static void mdlDerivatives(SimStruct *S)
  {
```

```
}
#endif /* MDL_DERIVATIVES */

/* 函数 mdlTerminate
 * 处理仿真结束时的一些工作 */
static void mdlTerminate(SimStruct *S)
{
}

#ifdef  MATLAB_MEX_FILE     /* 该文件被编译为 MEX 文件? */
#include "Simulink.c"       /* MEX 文件接口机制*/
#else
#include "cg_sfun.h"        /* 代码产生注册函数*/
#endif
```

为了节省篇幅，这里仅以例 6.12 的 Simulink 连续离散混合系统模型框架为例，讲述用 C MEX 文件编写 S-函数的方法。

例 6.13 用 C MEX 文件编写连续离散混合系统的 S-函数模块。

通过 C MEX 模板文件 sfuntmpl_basic.c 实现的 S-函数模块如下：

```
/* 文件名：exa07_504.c
 * 利用 C MEX 模板文件编写 S-函数，实现单位延迟（1/z）积分（1/s）的连续离散混合系统 */

#define S_FUNCTION_NAME mixedm
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"

#define U(element) (*uPtrs[element])   /* Pointer to Input Port0 */

/* S-函数方法 */

/* 函数 mdlInitializeSizes
* Simulink 利用该信息决定 S-函数模块的特性（如输入、输出和状态等变量数目等） */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);   /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
    }

    ssSetNumContStates(S, 1);
    ssSetNumDiscStates(S, 1);
    ssSetNumRWork(S, 1);   /* 对积分结果进行延迟操作 */

    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 1);
    ssSetInputPortDirectFeedThrough(S, 0, 1);
    ssSetInputPortSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetInputPortOffsetTime(S, 0, 0.0);
```

```
        if (!ssSetNumOutputPorts(S, 1)) return;
        ssSetOutputPortWidth(S, 0, 1);
        ssSetOutputPortSampleTime(S, 0, 1.0);
        ssSetOutputPortOffsetTime(S, 0, 0.0);

        ssSetNumSampleTimes(S, 2);

        ssSetOptions(S, (SS_OPTION_EXCEPTION_FREE_CODE |
                         SS_OPTION_PORT_SAMPLE_TIMES_ASSIGNED));

    } /* 结束 mdlInitializeSizes */

    /*函数 mdlInitializeSampleTimes
     * 指定 S-函数的连续和离散系统的采样时间都为 1.0 */:
    static void mdlInitializeSampleTimes(SimStruct *S)
    {
        ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
        ssSetOffsetTime(S, 0, 0.0);

        ssSetSampleTime(S, 1, 1.0);
        ssSetOffsetTime(S, 1, 0.0);
        ssSetModelReferenceSampleTimeDefaultInheritance(S);
    } /* 结束 mdlInitializeSampleTimes */

    #define MDL_INITIALIZE_CONDITIONS
    /*函数 mdlInitializeConditions
     * 初始化连续和离散系统的状态为 1 */
    static void mdlInitializeConditions(SimStruct *S)
    {
        real_T *xC0 = ssGetContStates(S);
        real_T *xD0 = ssGetRealDiscStates(S);

        xC0[0] = 1.0;
        xD0[0] = 1.0;

    } /* 结束 mdlInitializeConditions */

    /* 函数 mdlOutputs
     * 计算输出 y = xD，同时更新积分器的内部输出结果 */
    static void mdlOutputs(SimStruct *S, int_T tid)
    {
        /* 更新积分器内部输出结果"zoh" */
        if (ssIsContinuousTask(S, tid)) {
            if (ssIsSpecialSampleHit(S, 1, 0, tid)) {
                real_T *zoh = ssGetRWork(S);
                real_T *xC   = ssGetContStates(S);
                *zoh = *xC;
            }
        }
```

```
    /* 计算 y=xD */
    if (ssIsSampleHit(S, 1, tid)) {
        real_T *y    = ssGetOutputPortRealSignal(S,0);
        real_T *xD   = ssGetRealDiscStates(S);
        y[0]=xD[0];
    }

} /* 结束 mdlOutputs */

#define MDL_UPDATE
/* 函数 mdlUpdate
 * 计算 xD = xC */
static void mdlUpdate(SimStruct *S, int_T tid)
{
    UNUSED_ARG(tid); /* not used in single tasking mode */

    /* 计算 xD=xC */
    if (ssIsSampleHit(S, 1, tid)) {
        real_T *xD = ssGetRealDiscStates(S);
        real_T *zoh = ssGetRWork(S);
        xD[0]=*zoh;
    }

} /* 结束 mdlUpdate */

#define MDL_DERIVATIVES
/* 函数 mdlDerivatives
 *  计算连续系统状态的导数 xdot = U */
static void mdlDerivatives(SimStruct *S)
{
    real_T              *dx    = ssGetdX(S);
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);

    /* 计算 xdot=U */
    dx[0]=U(0);

} /* 结束 mdlDerivatives */

/* 函数 mdlTerminate
 * 结束仿真 */
static void mdlTerminate(SimStruct *S)
{
    UNUSED_ARG(S); /* 没有使用的输入参数 */
}

#ifdef  MATLAB_MEX_FILE     /* 该文件被编译为 MEX 文件吗? */
#include "Simulink.c"        /* MEX 文件的接口机制 */
#else
#include "cg_sfun.h"         /* 代码产生注册函数 */
#endif
```

6.5.5 S-函数创建器的使用

我们知道，依照前面方法设计出的 S-函数虽然可以带有附加参数，但模型参数输入较不方便。因为需要在附加参数栏目中同时输入若干个参数，没有任何提示，所以，可以用封装模块的方法来封装该 S-函数，设计出相应的参数输入对话框。

Simulink 提供了一种自动生成 C 语言 S-函数的工具——S-Function Builder（S-函数创建器），利用它可以编写简单的 C 语言 S-函数。S-Function Builder 可以根据用户的各种设置自动生成 C 语言 S-函数的框架和代码，通过它用户不需要重新构造函数的框架。但是它在一定程度上限制了 C 语言 S-函数的功能，因为通过 S-Function Builder 构造的 S-函数只能有一个输入信号和一个输出信号，而且只能处理 double 类型的数据。

要使用 S-Function Builder 创建一个 S-函数，应该遵循以下规则与步骤：

（1）把 Matlab 的当前路径设置在想要保存 S-函数的路径上。

（2）创建一个新的 Simulink model 文件。

（3）从 Simulink 用户自定义函数库复制一个 S-Function Builder 模块到新建的 model 文件中。

（4）双击这个模块打开 S-Function Builder 参数设置对话框，如图 6-92 所示。

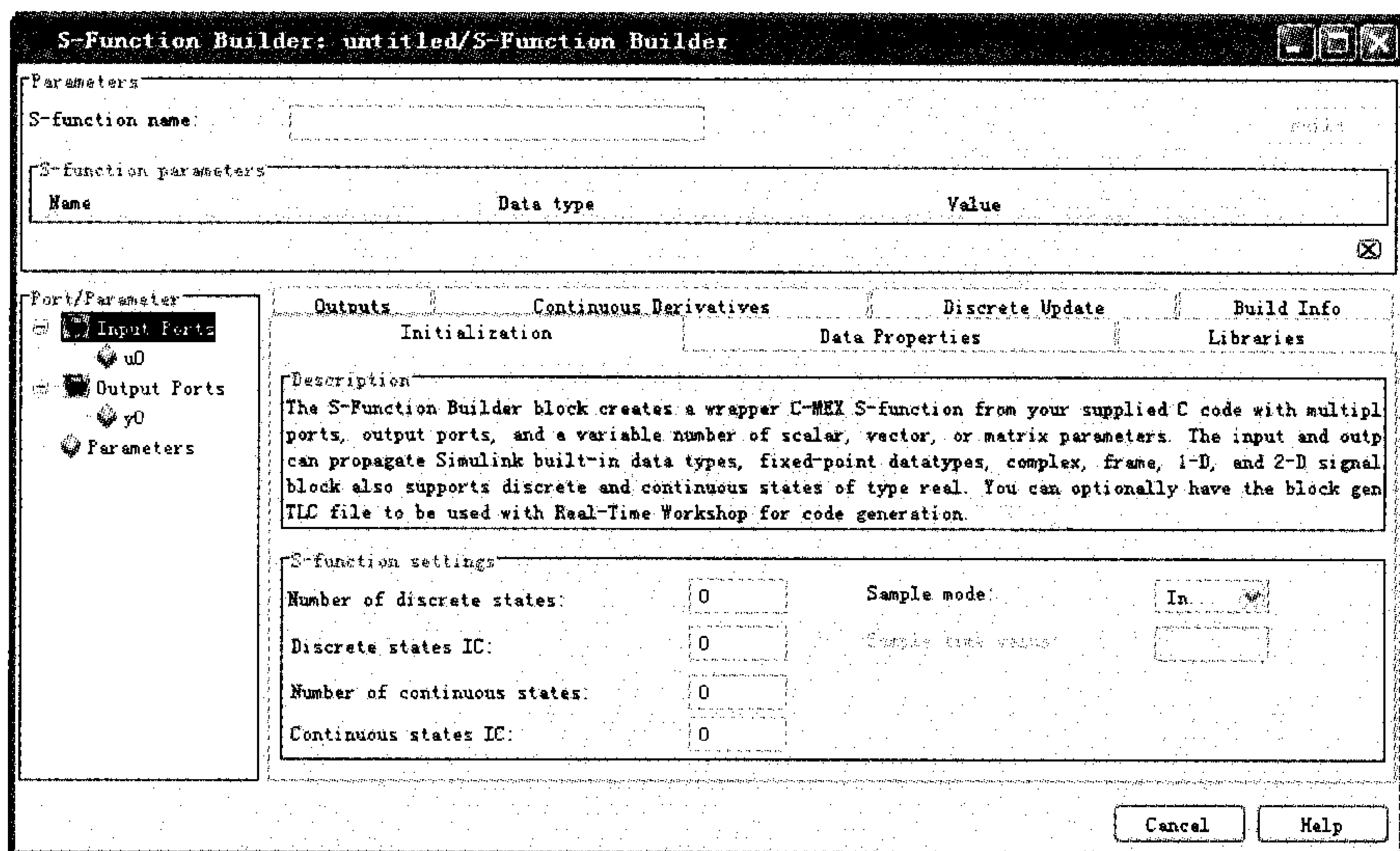


图 6-92　S-Function Builder 参数设置对话框

（5）在 S-函数名字域写入想创建的 S-函数的名字。

（6）如果这个 S-函数还有参数，则在 S-函数参数域写入这些参数的默认值。

（7）根据 S-Function Builder 的书写规范写入一些所需的源程序代码。

（8）如果还没有配置 mex 命令，则要在系统中先配置好 Matlab mex 命令。配置 mex 命令的方法是在 MATLAB 命令窗中输入 mex –setup 指令。

（9）单击 S-Function Builder 对话框中的【Build】按钮开始创建 S-函数的过程。Simulink 会创建一个执行特殊定义的 S-函数 MEX 文件，并且把它存在当前目录下。

（10）保存这个包含有 S-Function 的模型文件。

1．S-Function Builder 创建 S-函数的方法

S-Function Builder 是按照如下方法建立 S-Function 的。首先，它在当前目录下产生下列源文件。

- sfun.c。sfun 是要创建的 S-函数的名字。此文件包含了 C 语言程序源代码，它可以代表生成的 S-函数的标准部分。
- sfun_wrapper.c。此文件包含了读者在 S-Function Builder 对话框里输入的传统代码。
- sfun.tlc。此文件使得 Simulink 可以以更快的模式运行创建的 S-函数，并且使 RTW 可以把这个 S-函数包含在它产生的代码中。

在产生 S-函数源代码以后，S-Function Builder 用 Matlab 的 mex 命令建立 mex 文件，接着便可以开始使用这个 S-函数了。

2．设置包含路径

S-Function Builder 会通过 MATLAB 的应用数据，即所谓的 SfunctionBuilderIncludePath 在指定的路径下寻找头文件。这个数据与用户创建的 S-Function Builder 模块的所在模型文件相关联。如果用户的 S-函数使用传统的头文件，但这些文件并不在当前目录下，则用户必须设置包含这些头文件目录的位置。SfunctionBuilderIncludePath 是一个三维元胞数组，利用它可以指定 3 个包含路径。设置方法如下：

```
>> incPath=getappdata(0, 'SfunctionBuilderIncludePath');
>> incPath{1}=' / home / jones / include';
>> incPath{2}: getenv('PROJECT_INCLUDE_DIR');
>> setappdata(0, 'SfunctionBuiiderIncludePath', incPath);
```

上述命令设置 SfunctionBuilderIncludePath 包含了两个路径。

3．S-Function Builder 的参数设置界面及其使用方法

用户可以在 S-Function Builder 参数设置界面中输入相应的信息和所需的传统代码。这里仅介绍创建器初始化、数据特征和代码输出页中的参数情况。

1）创建器初始化

创建器初始化页面如图 6-93 所示，用户可以通过该页面输入 S-函数的基本特征信息，例如，输入/输出端口的宽度和抽样时间等。

图 6-93　初始化页面的参数设置

S-Function Builder 借助用户在这个界面的输入信息，生成 S-函数的 mdlInitializesizes 调用函数。初始化页面包含以下参数。

- 离散状态数量（Number of discrete states）。
- 离散状态初始条件（Discrete states IC）。
- 连续状态数量（Number of continuous states）。
- 连续状态初始条件（Continuous states IC）。
- 抽样模式（Sample mode）。
- 抽样时间值（Sample time value）。
- 输入端口宽度（Input port width）。
- 输出端口宽度（Output port width）。
- 参数个数（Number of parameters）。

2）数据特征

在数据特征页面中用户可以为自己定义的 S-函数设置输入/输出端口及相关参数，如图 6-94 所示。

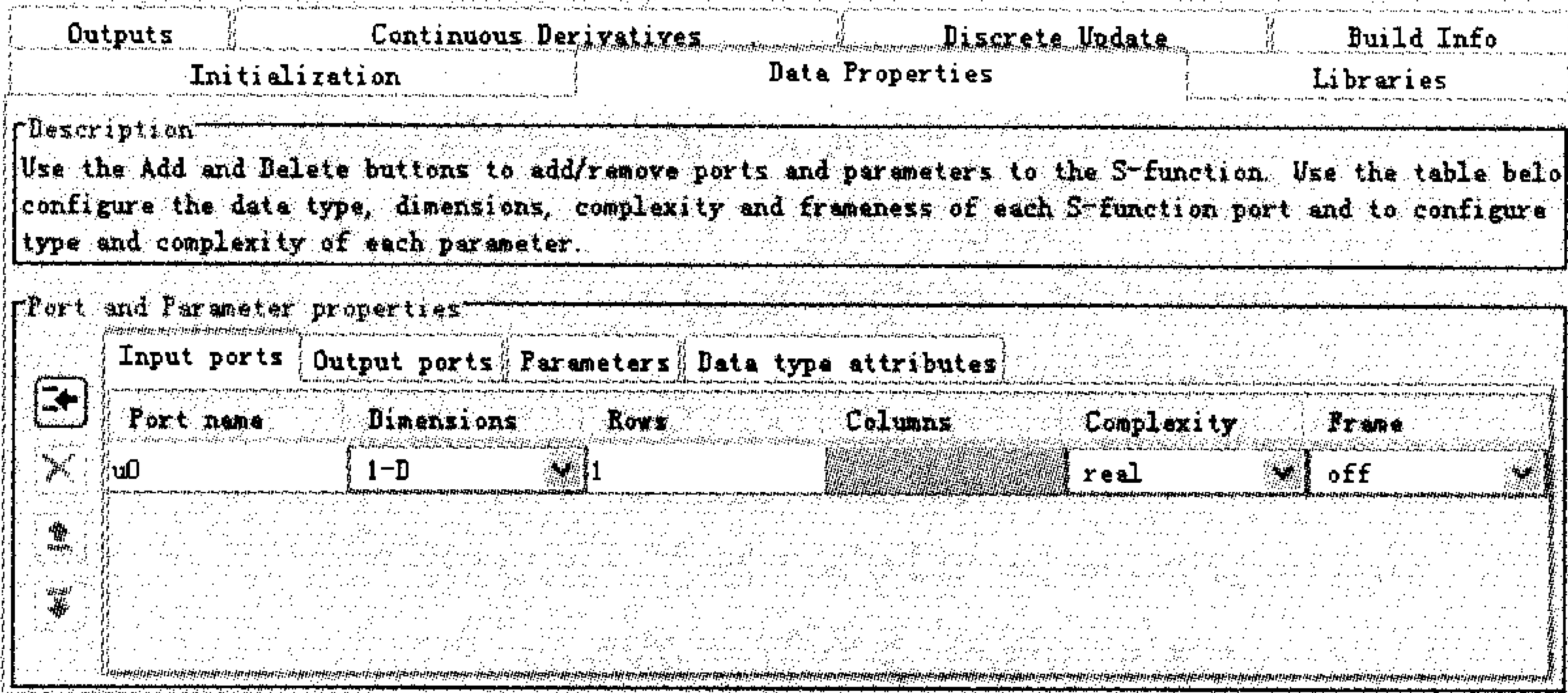


图 6-94　数据特征页面的参数设置

这个页面又分为四大部分：输入端口、输出端口、参数和数据类型属性。但输入/输出端口选项需要输入的信息是基本一致的：端口名字（Port Nname）、端口维数（Dimensions）、行数（Rows）、列数（Columns）、是否为复数信号（Complexity），以及数据是否为帧格式（Frame）；参数选项需要输入的信息是参数的名称（Parameters Name）、数据类型（Data Type）和是否为复数（Complexity）；数据类型属性显示输入/输出端口的数据类型（Data Type）、字节长度（Word Length）、分区长度（Franction Length）、范围（Slope）和偏移量（Bias）等信息。

3）代码输出

用户可以在如图 6-95 所示的代码输出页面中输入 S-函数 outputs 代码。当 S-Function Builder 生成 S-函数代码的时候，会将这些代码写在下面 wrapper 函数中“用户写入的代码”位置。

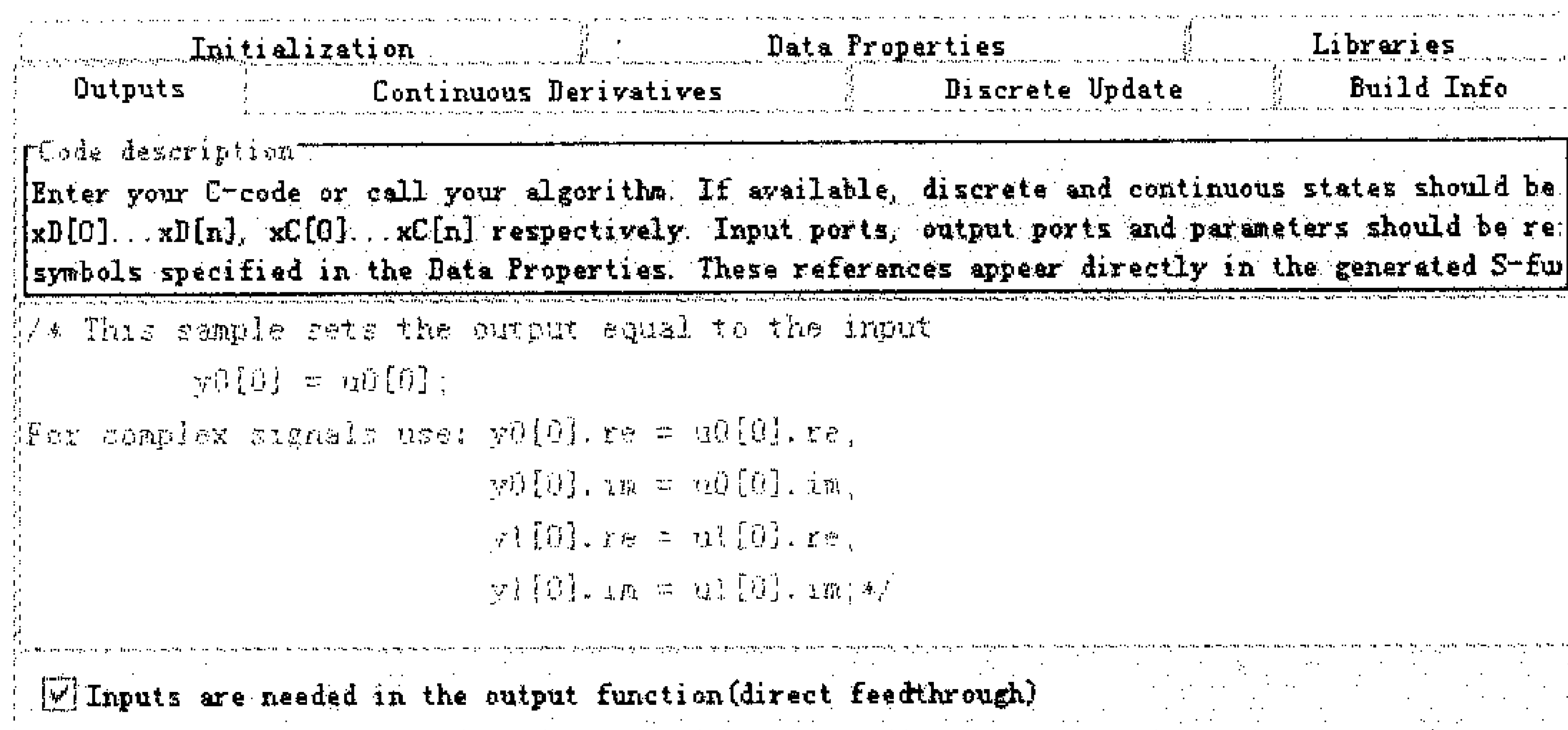


图 6-95　输出页面的代码设置

```
void sfun_Outputs_wrapper(const real_T*u,
                          real_T *Y,
                    const real_T *xD,        %可选
                    const real_T *xC,        %可选
                    const real_T *param0,    %可选
                          int_T p_width0,    %可选
                          real_T *paraml,    %可选
                          int_t p_widthl,    %可选
                          int_T y_width,     %可选
                          int_T u_width)     %可选
{
    % 用户写入的代码
}
```

4. S-Function Builder 实例演示

例 6.14 利用 S-Function Builder 功能实现对输入信号放大 x 倍（x 是这个 S-函数的参数）。首先，创建如图 6-96 所示的 Simulink 模型。

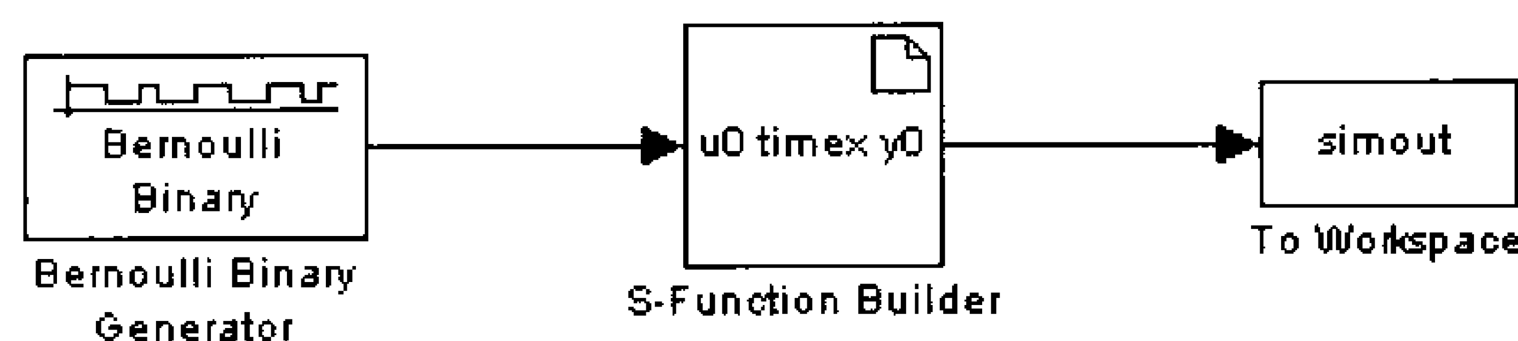


图 6-96　S-Function Builder 创建 S-函数的模型

其次，双击“S-Function Builder”图标，在打开对话框的 S-function name 文本框中输入 timex，在 S-function parameters 的下面输入参数的名字、数据类型和默认值，如图 6-97 所示。在输入 S-function parameters 相关内容时，首先需要在数据特性（Date properties）的 Parameter 栏中添加 TIMES 参数。

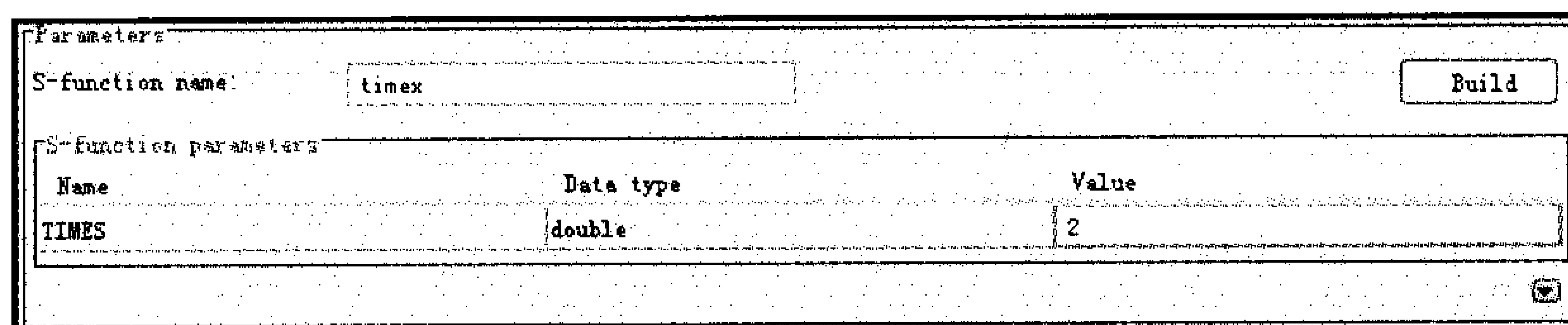
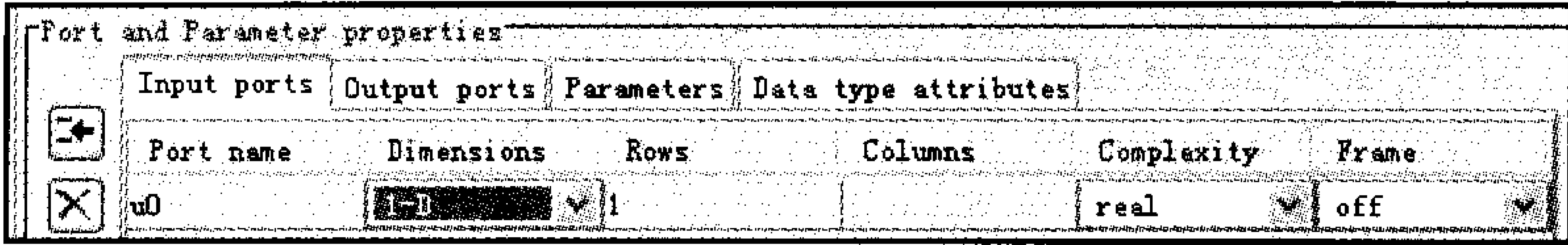
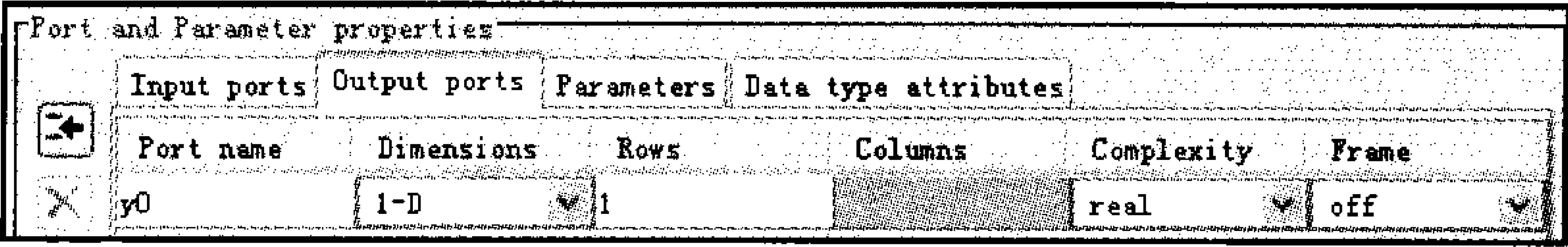


图 6-97　S-函数的名称与参数设置

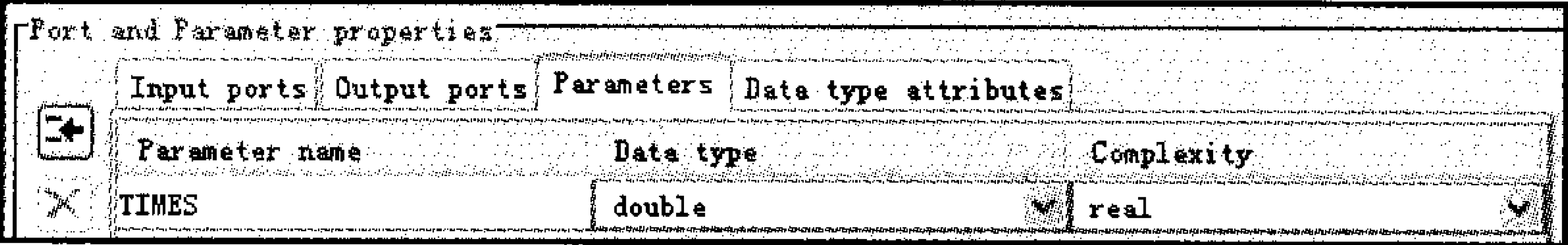
其次，打开数据特征页面输入输入端口、输出端口和参数的相关特征信息，如图 6-98 所示。



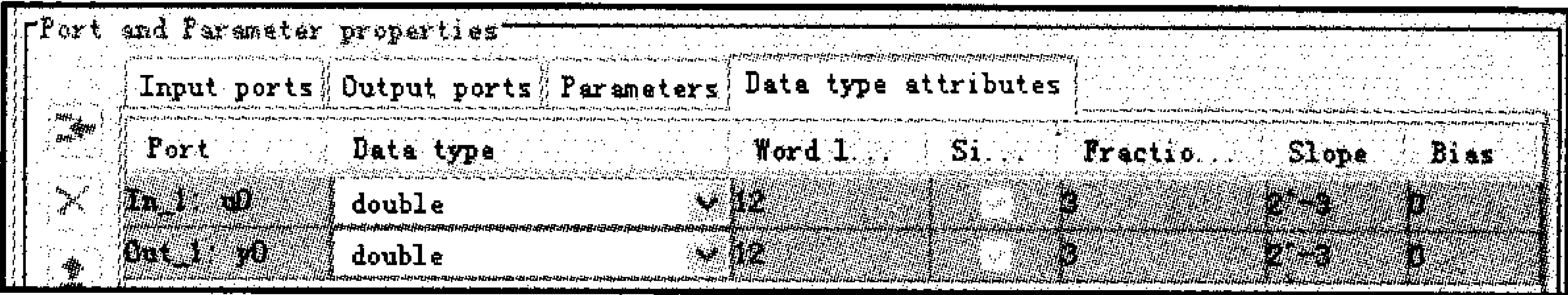
（a）输入端口参数设置



（b）输出端口参数设置



（c）系统参数设置



（d）数据类型属性设置

图 6-98 数据特征参数设置

再次，打开代码输出页面，输入下列代码，如图 6-99 所示。

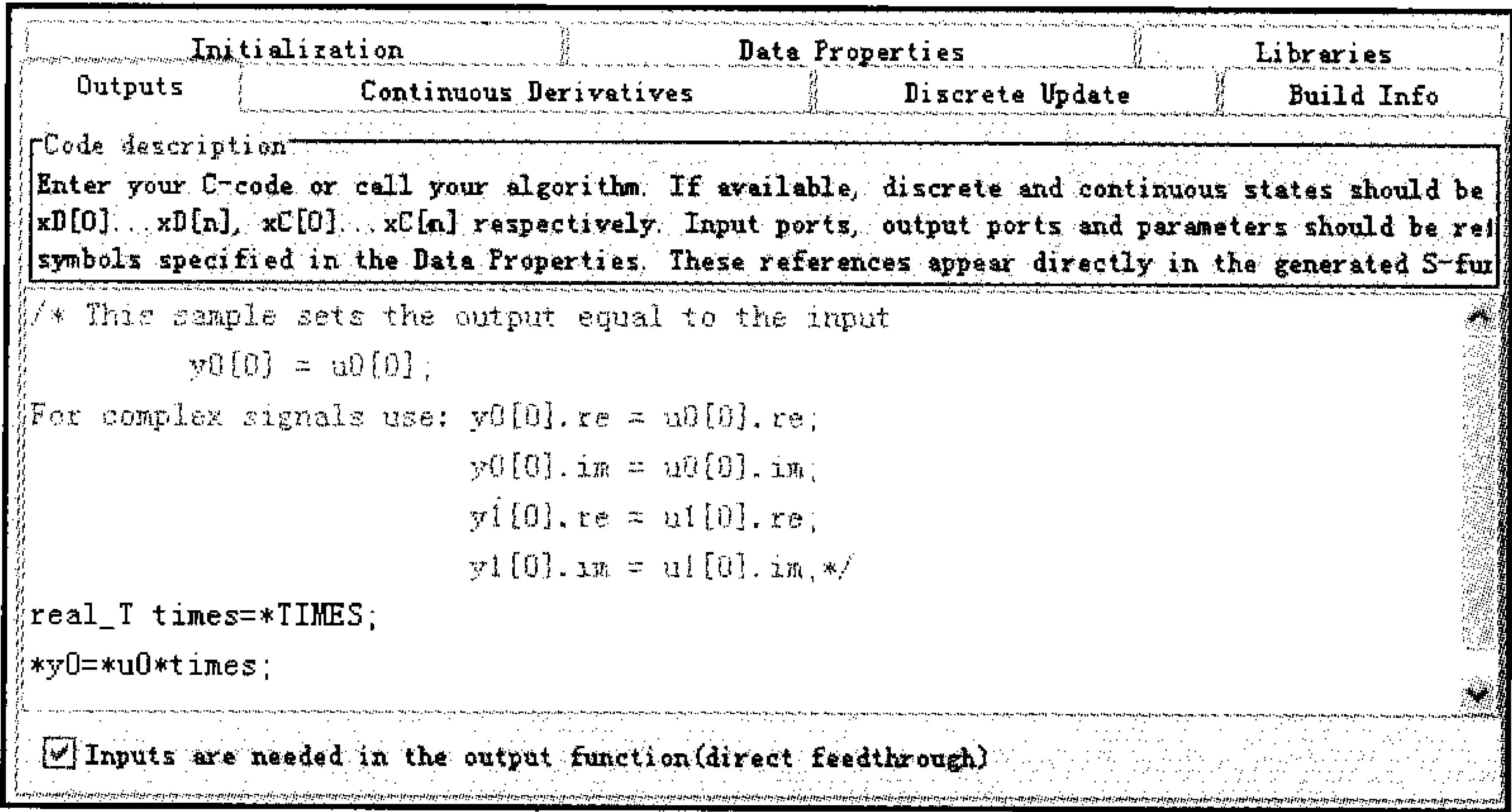


图 6-99 代码输出页面输入的代码

```
real_T times=*TIMES;
*y0=*u0*times;
```

最后，单击【Build】按钮开始创建过程，创建成功之后将在 Build Iinfo 页面显示相关

信息，如图 6-100 所示。之后就可以通过如图 6-96 所示的 Simulink 模型进行仿真了。

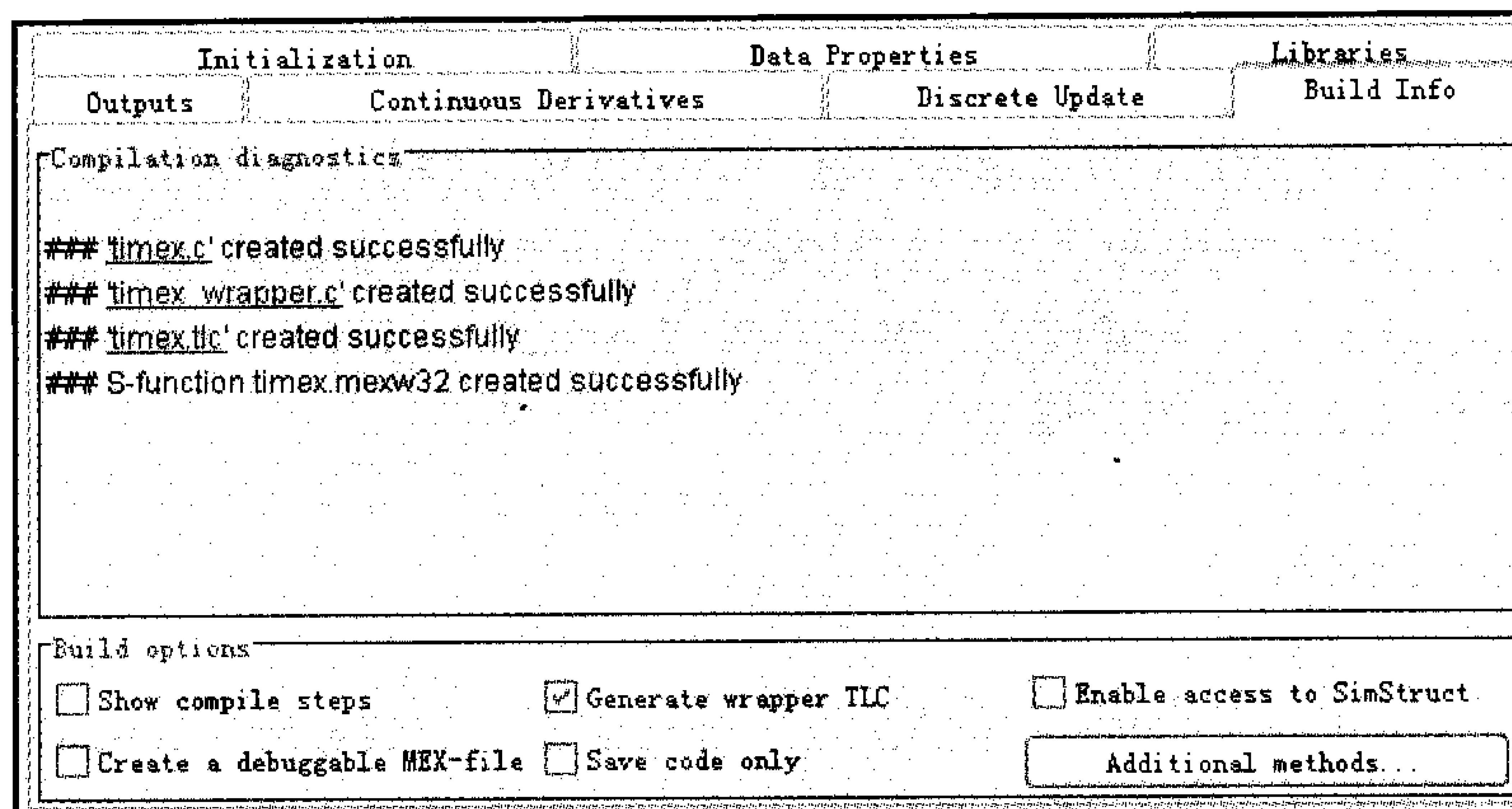


图 6-100　S-Function Builder 代码创建过程信息

生成的 timex_wrapper.c 的程序代码如下所示，读者通过这个程序代码可以很清楚地了解到自己所写的代码是如何被嵌入的。

```
/* 该文件由 S-FUNCTION BUILDER: 3.0 产生
  *     "Wrapper S-functions"    */

/* 所包含的文件 */
#if defined(MATLAB_MEX_FILE)
#include "tmwtypes.h"
#include "simstruc_types.h"
#else
#include "rtwtypes.h"
#endif
#include <math.h>
#define u_width 1
#define y_width 1

/* 产生外部参考 */
/* 输出函数 */
void timex_Outputs_wrapper(const real_T *u0,
                           real_T *y0,
                            const real_T   *TIMES, const int_T p_width0)
{
         y0[0] = u0[0];
For complex signals use: y0[0].re = u0[0].re;
                          y0[0].im = u0[0].im;
                          y1[0].re = u1[0].re;
                          y1[0].im = u1[0].im;*/
/* 在输出代码窗口输入的两行代码 */
real_T times=*TIMES;
*y0=*u0*times;
}
```

6.6 Stateflow 原理与应用

Stateflow 是有限状态机的图形实现工具，它可以用于解决复杂的监控逻辑问题，用户可以用图形化的工其来实现各个状态之间的转换。可以在 Simulink 中直接嵌入 Stateflow 生成的监控逻辑，达到两者的无缝连接。本章讲述了 Stateflow 原理、应用基础和常用命令，并通过实例来说明 stateflow 的建模方法。

6.6.1 Stateflow 原理

Stateflow 和 Simulink 环境可以综合在一起进行建模、仿真和分析。Stateflow 可以通过图形环境来设计监视控制系统。通过有限状态机的图形化表示，通常可为一个相对复杂的控制系统的建模和仿真提供一个清晰明了的描述。能够将系统的描述和设计结合得更为紧密，更加容易设计，利于考虑各种情况，可以不断修改，直到设计满足要求为止。

Stateflow 仿真的原理是有限状态机（Finite State Machine，FSM）理论，所谓有限状态机，就是指在系统中有可数的状态，在某些事件发生时，系统从一个状态转换成另一个状态，所以有限状态机系统又称为事件驱动的系统。在有限状态机的描述中，可以设计出从一个状态到另一个状态转换的条件，在每对相互可转换的状态下都设计出状态迁移的事件，从而构造出状态迁移图。

在 Stateflow 中提供了图形界面支持的设计有限状态机的方法，它允许用户建立起有限的状态，并用图形的形式绘制出状态迁移的条件，从而构造出整个有限状态机系统。所以在 Stateflow 下，状态和状态转换是其最基本的元素。

Stateflow 模型一般是嵌在 Simulink 模型下运行的，Stateflow 图是事件驱动的，这些事件可以来自相同 Stateflow 图，也可能来自 Simulink 模型。事实上，在仿真初始化过程中，Simulink 将自动启动编译程序，将 Stateflow 绘制的逻辑框图变换为 C 格式的 S-函数，从而在仿真过程中直接调用相应的动态连接库文件，将二者构成一个仿真整体。

6.6.2 Stateflow 应用基础

在 Simulink 中，Stateflow 利用图形来表示离散模型集合的。它通过改变控制目标状态这一事件，激发 Stateflow 有限状态机的运行。为了让读者掌握 Stateflow 在 Simulink 模型中的应用方法，这里主要讲述对象状态的 Stateflow 表示，以及 Stateflow 选择控制目标，以及使用数据变量等知识。

1. Chart 模型的编辑环境

Stateflow 是 Simulink 的一种工具，它用于控制 Simulink 模型中目标的状态。这个目标可以是发动机、水泵等设备。

进入 Stateflow 编辑环境的方法是在 MATLAB 命令窗口中输入指令

```
>> stateflow
```

将弹出如图 6-101 所示的 Stateflow 库窗口。

双击图 6-101 中的 Chart 模型，将得到如图 6-102 所示的窗口，用户可以在此窗口中编辑所需要的 Stateflow 目标控制模块。Stateflow 提供了强大的框图编辑功能，可以描述非常复杂的逻辑模型。

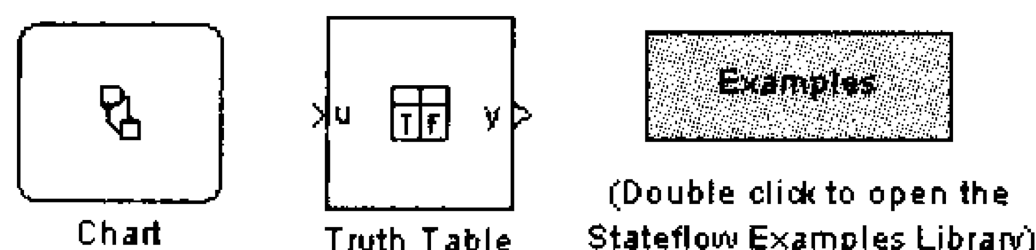


图 6-101　Stateflow 的模块库

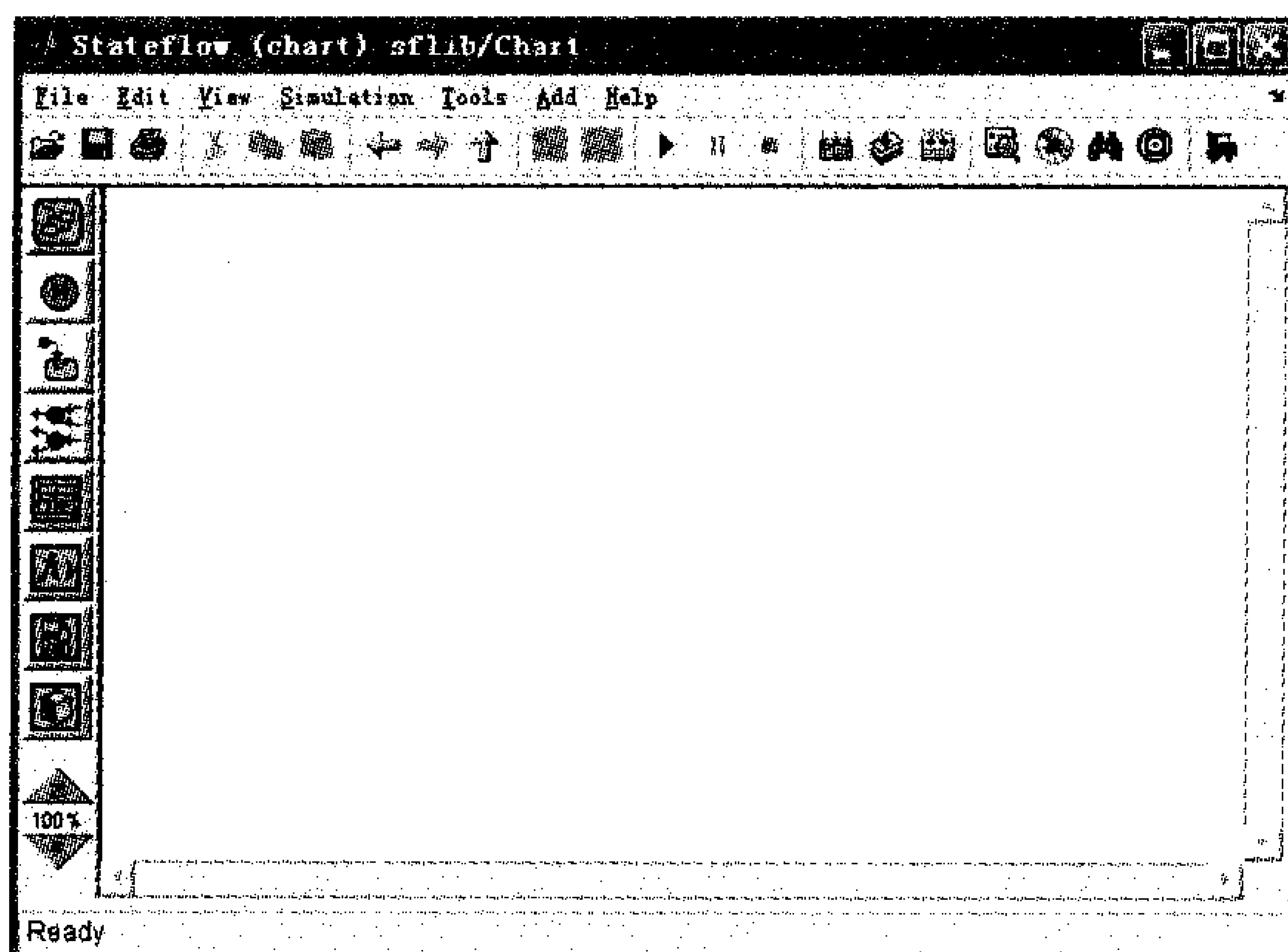


图 6-102　Stateflow Chart 编辑窗口

在图 6-102 所示的 Chart 模型编辑窗口中，左侧工具栏中的工具从上至下依次是“状态”、“历史交汇”、“默认迁移”、“连接交汇”、“真值表”、“函数”、“嵌入式 MATLAB 函数”和“盒子”；窗口上方工具栏前面部分的工具我们都比较熟悉，这里主要介绍这 8 个工具，它们从左至右依次是“模型重新全部创建”、“剖析对话框”、“模型创建”、“模型信息浏览”、“调试”、“查找”、“仿真目标”和“Simulink 库浏览器”。

如果需要改变 Chart 模型窗口的属性，用户可以在 Stateflow 编辑界面空白处单击鼠标右键，在弹出的快捷菜单中执行【Properties】（属性）指令，则会弹出如图 6-103 所示的对话框。用户可以设置 Chart 模型的属性包括：状态机类型、更新方法、采样时间、C-位操作、指定状态/迁移的顺序、输出 Chart 模型级图形函数、使用 Simulink I/O 的数据类型、初始化时执行 Chart 模型、每次激活 Chart 模型时初始化输出、断点设置和模型描述。

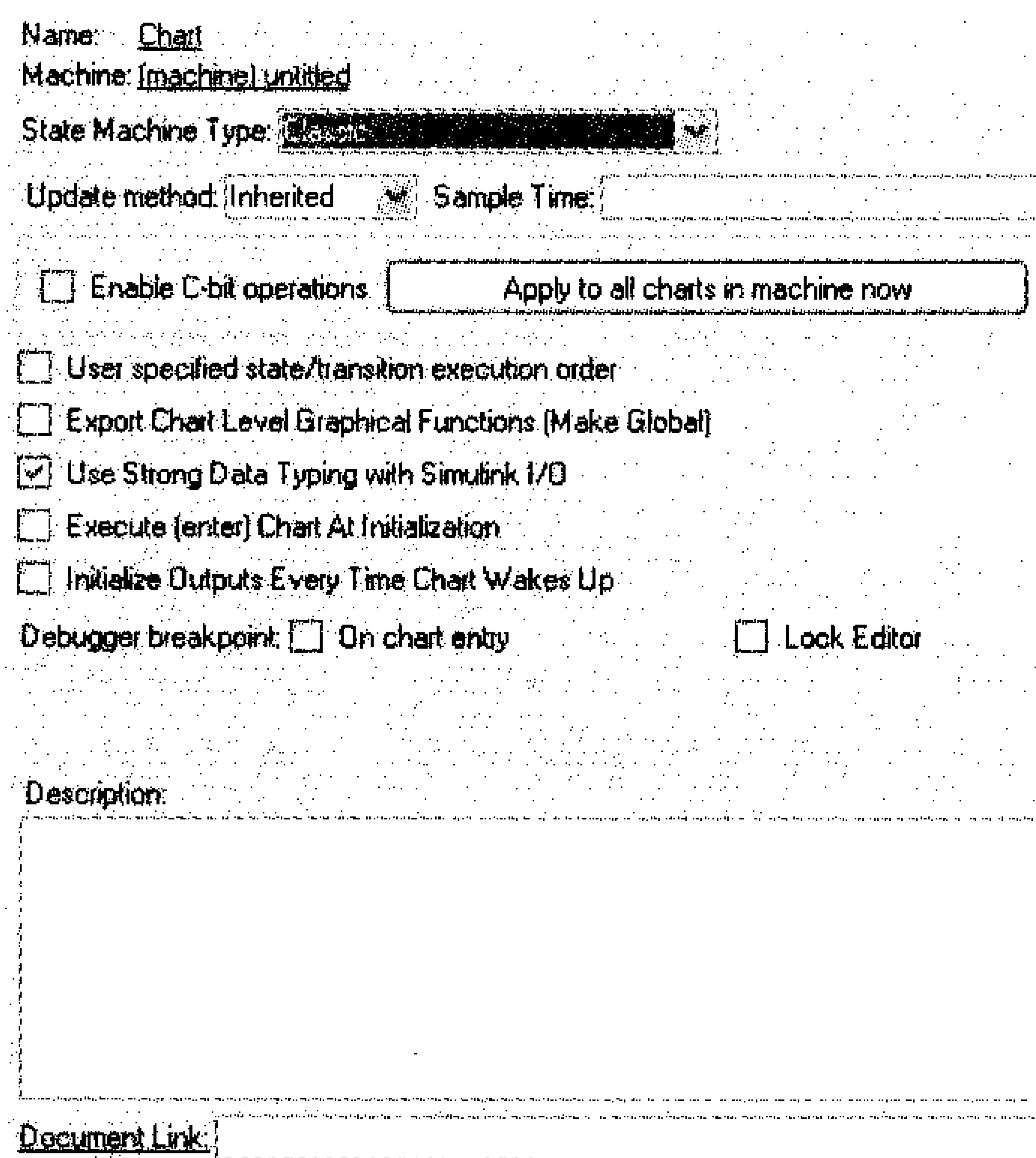


图 6-103　Stateflow Chart 模型的属性设置对话框

2．对象状态的建立

对象状态分为激活（Active）和非激活（Inactive）两种。通过图 6-102，单击【状态】按钮，用户可以在图形编辑框窗中添加对象的状态模块。如图 6-104 所示，我们建

立了对象 on 和 off 的两种状态。

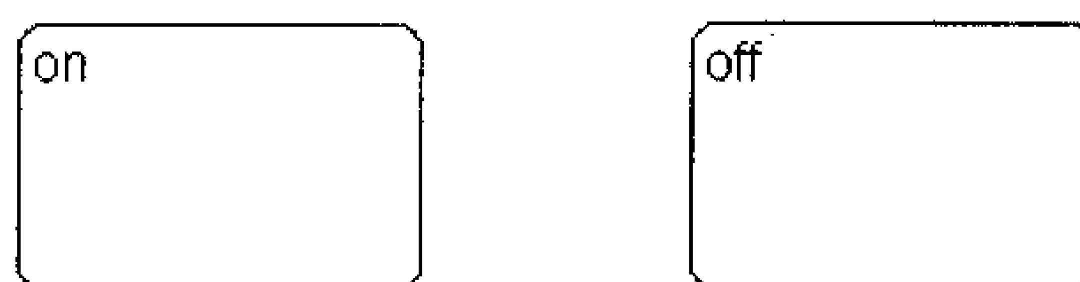


图 6-104　on 和 off 对象状态模块

同时我们还可以设置状态模块的属性，方法是选中状态模块，单击鼠标右键，在弹出式菜单中执行【Properties】指令，那么则会弹出图 6-105 所示的状态模块属性设置对话框。通过该对话框，我们可以设置状态模块的断点、标签和模块描述等参数。

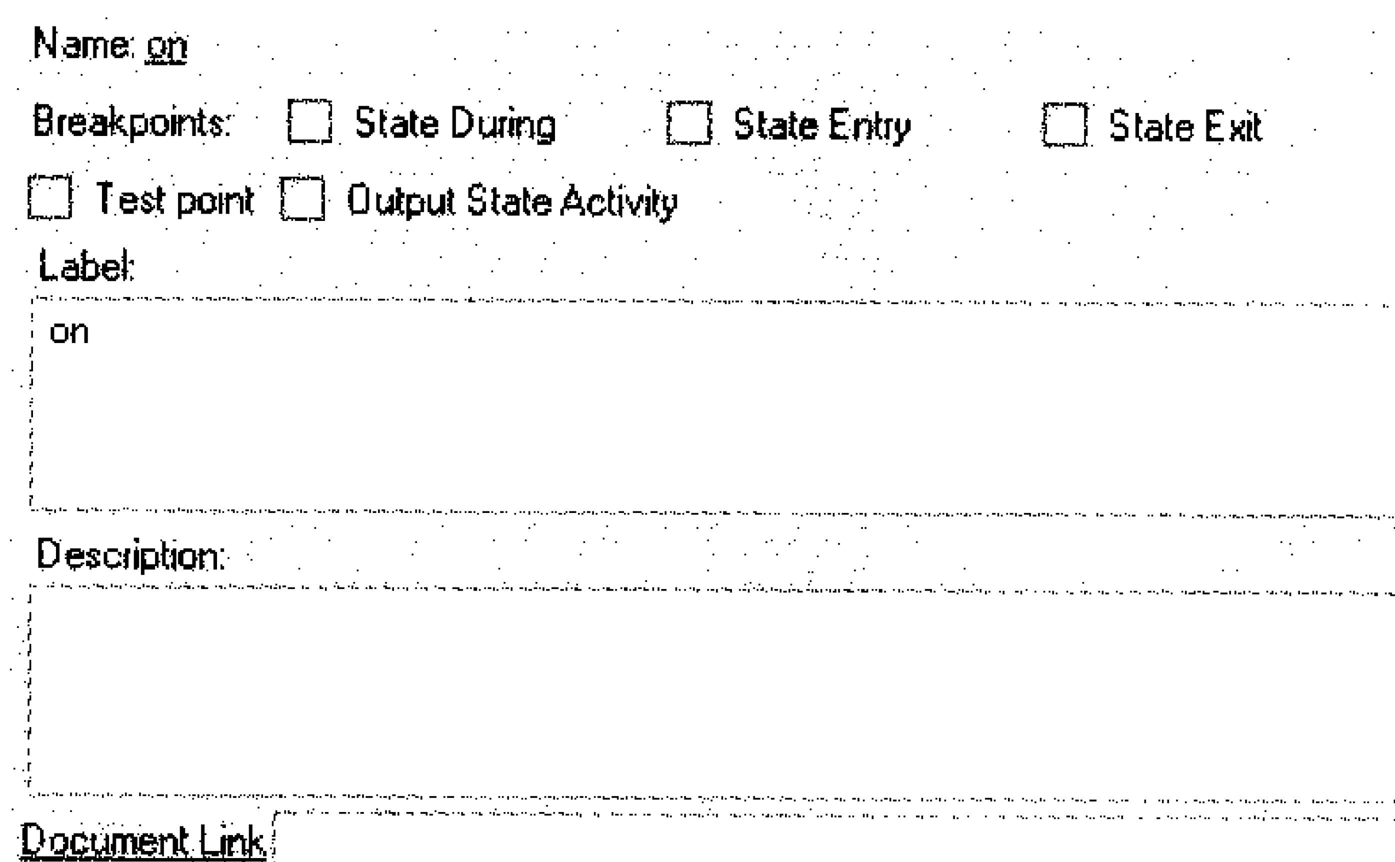


图 6-105　on 对象状态模块的属性设置对话框

3．对象状态的迁移

为了使模型变得更加灵活，对象需要具有状态迁移的功能。针对 on 和 off 对象状态，我们设置 3 种状态迁移方式：默认迁移方式，off 状态迁移至 on 状态方式和 on 状态迁移至 off 状态方式，如图 6-106 所示。

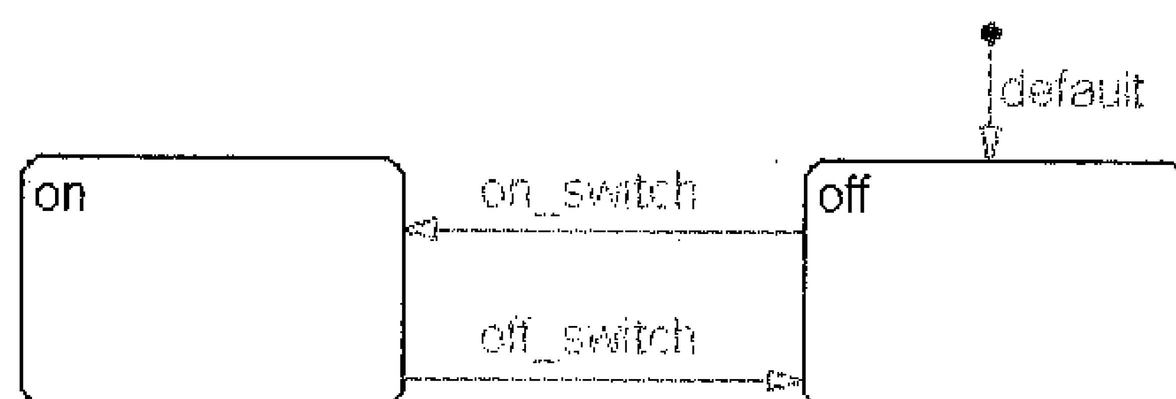


图 6-106　对象状态的迁移

具体的状态迁移设置方法为：单击工具栏中的【Default transition】按钮，接着使状态迁移指在一个状态的边界，然后拖动迁移的另一端至另外一个状态边界处释放，则可以绘制出从一个状态转换成另外一个状态的连线。对于状态标签的填写，可以通过双击状态连线或打开状态属性设置对话框来实现。

对象状态迁移起始于源状态，终止于目标状态。若源状态被激活后，并执行了迁移，那么，源状态就会变成非激活状态，而目标状态在会被激活。例如，off 处于激活状态，则通过 on_switch 迁移，on 的状态就会被激活，而 off 则处于非激活状态。但若 on 的状态处于激活状态，却不能通过 on_switch 迁移来使 off 状态被激活。如果要实现这个目标，就必须采用 off_switch 迁移。

默认迁移没有明显源状态，它是一类比较特殊的迁移。默认状态可以是激活或者非激活状态，但指向 off 状态，当 Stateflow 图形激活时，它也将随之激活。

4．对象状态的激发

对象的状态可通过事件（如温度、水准点和开关等）来激发。通常，事件是通过 Simulink 中建立的实际模型来产生的，然后触发相应的 Stateflow 模块做出反应。

如果没有事件的驱动，Stateflow 图形不会被激活，状态间的迁移也不会发生。在图 6-106 中，当且仅当 off 处于激活状态，并且收到 on_switch 事件，激活才能实现由状态 off 迁移到状态 on。同样，当且仅当 on 处于激活状态，并且收到 off_switch 事件，激活才能实现由状态 on 到状态 off 的迁移。

添加事件可以通过执行【Add】→【Event】命令，其中有“Local（局部）”“Input from Simulink（从 Simulink 输入）”和“Output to Simulink（输出到 Simulink）”3 个选项，如果执行“Input from Simulink”命令，将弹出如图 6-107 所示的参数设置对话框。在该对话框中，我们可以设置事件的名称，范围、端口、触发模型和调试断点等参数。图 6-106 中的 on_switch 和 off_switch 是局部事件。

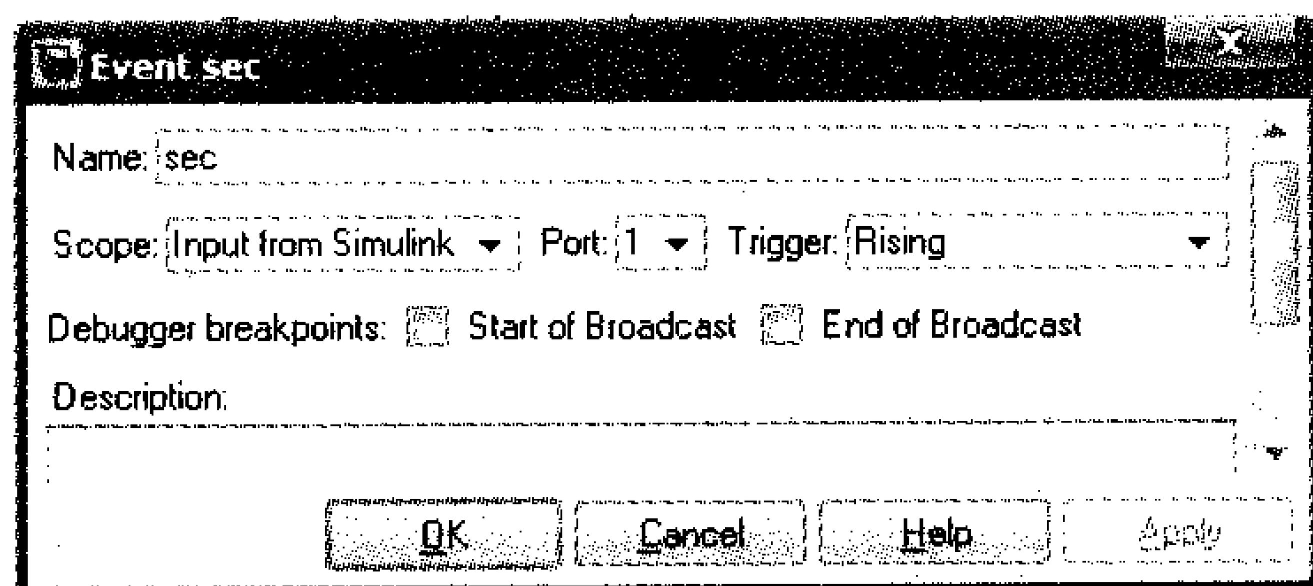


图 6-107 事件及其参数设置对话框

5．对象状态的连接

对于对象的状态，有时并不希望事件一旦激发，便产生状态的迁移。此时，我们需要通过连接提供决策点，决定事件是否需要迁移。决策逻辑可以采用 for 循环和 if-then-else 结构，以流程图符号来更加准确地描述模块状态。

对于图 6-106，我们在 off-on 迁移中再增加连接。方法是单击【Connective Junction】按钮，放置连接点，然后通过该连接点进行连线，结果如图 6-108 所示。

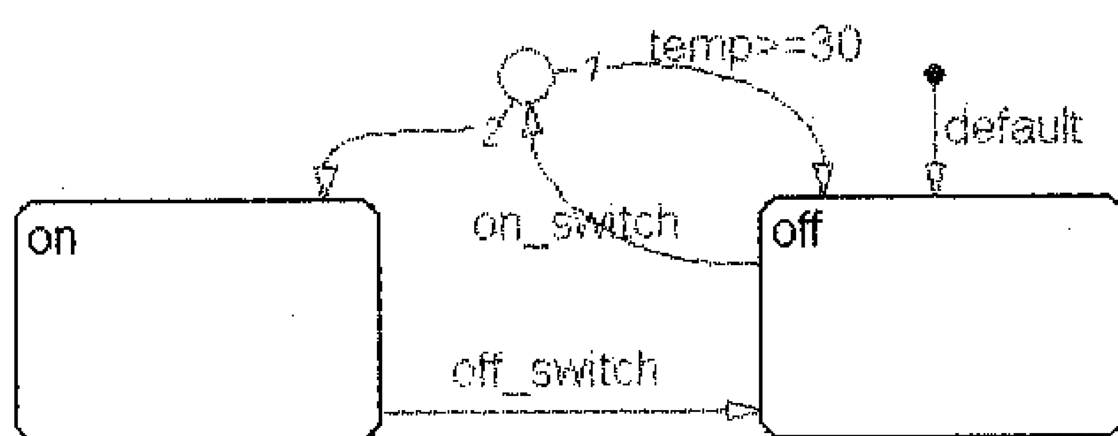


图 6-108 状态 off 到状态 on 的连接

在图 6-108 中，如果温度大于或等于 30℃时，状态 off 就会重新返回状态 off，这样状态 off 仍然处于激活状态。如果温度小于 30℃，那么状态 off 就会便被迁移到状态 on，这时状态 on 处于激活状态，状态 off 处于非激活状态。

6．数据变量的定义

Stateflow 与 Simulink 交流是通过事件和数据变量进行的。事件的添加方法我们在前文已经讲述，现在我们讲解数据变量的添加方法。这里我们添加 reference 和 temp 两个从 Simulink 输入的数据变量，同时添加 LED 和 boiler 两个输出到 Simulink 的数据变量。方法如下：

执行【Add】→【Data】命令，其中除了有"Local（局部）""Input from Simulink（从 Simulink 输入）"和"Output to Simulink（输出到 Simulink）"3 个选项外，还有"Constant（常量）"、"Parameter（参数）"和"Data Store Memory（数据存储记忆）"3 个选项。参数设置如图 6-109 所示。在该对话框中，我们可以设置数据变量的名称、范围、端口、大小和数据类型模式等参数。

设置好输入/输出数据变量的 Chart 模块如图 6-110 所示。

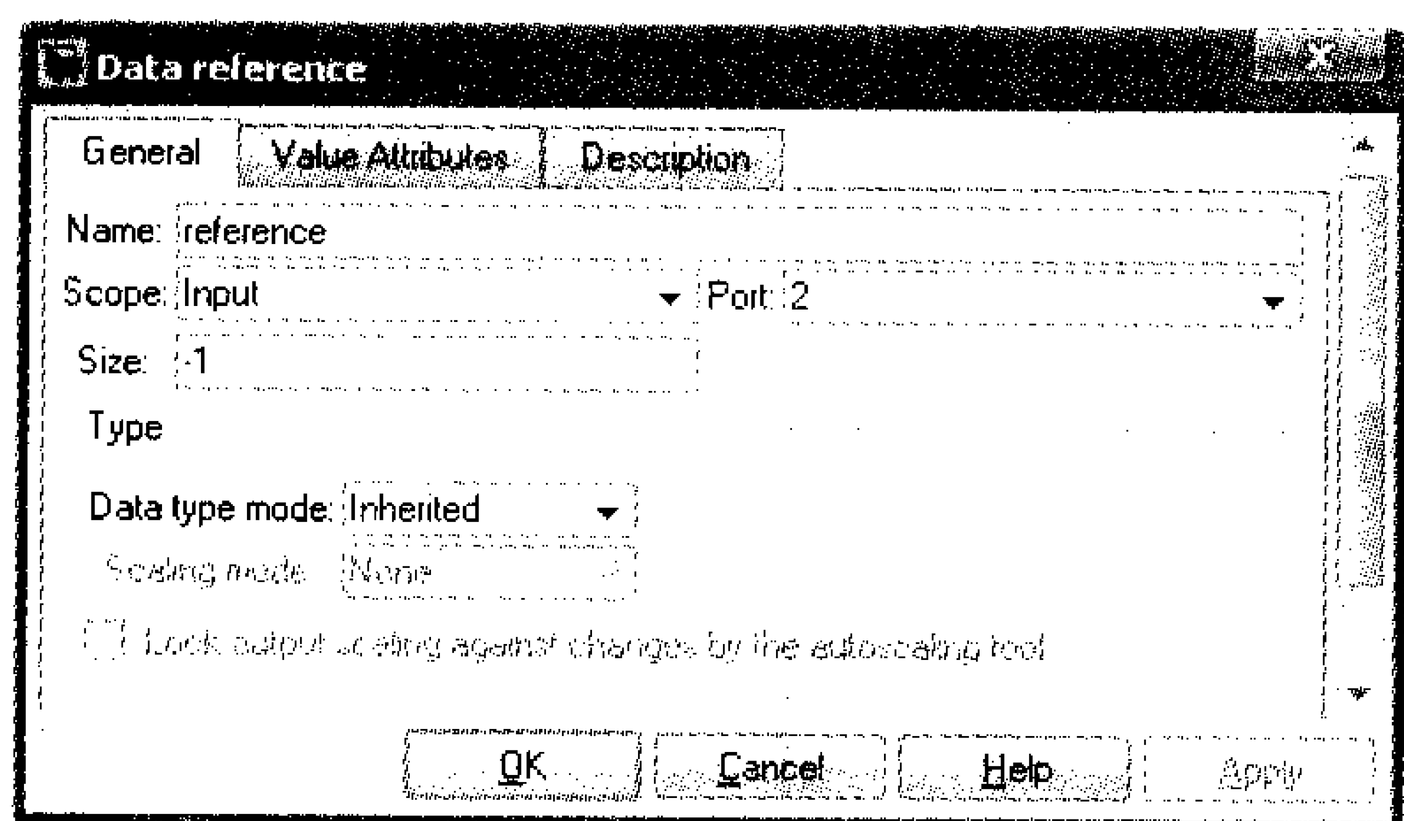


图 6-109　数据变量的设置

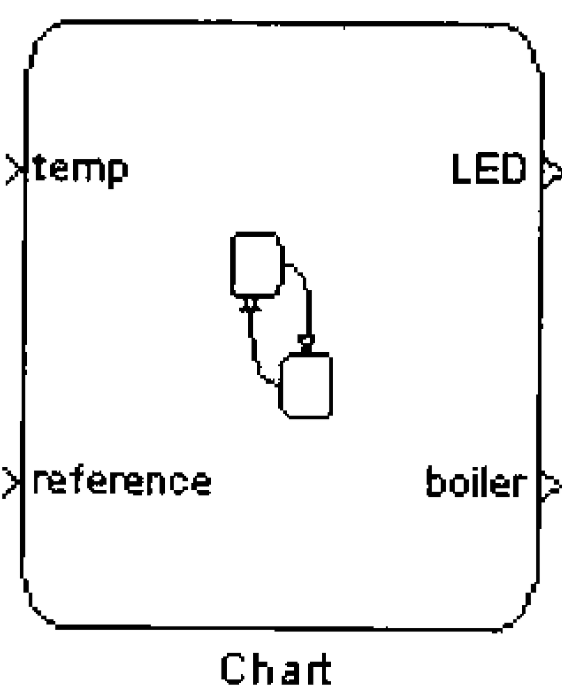


图 6-110　定义输入/输出数据变量的 Chart 模块

7．事件和数据变量的管理

在 Stateflow 中，有时需要对事件和数据变量进行管理，比如删除多余的事件或数据变量。此时，我们可以执行【View】→【Model Explorer】（模型浏览器）命令，在弹出的模型浏览器中通过 Chart 选项进行管理，如图 6-111 所示。此时我们可以通过 Chart 内容栏，选中欲管理的参数，并通过弹出菜单或键盘按键来完成相应操作。

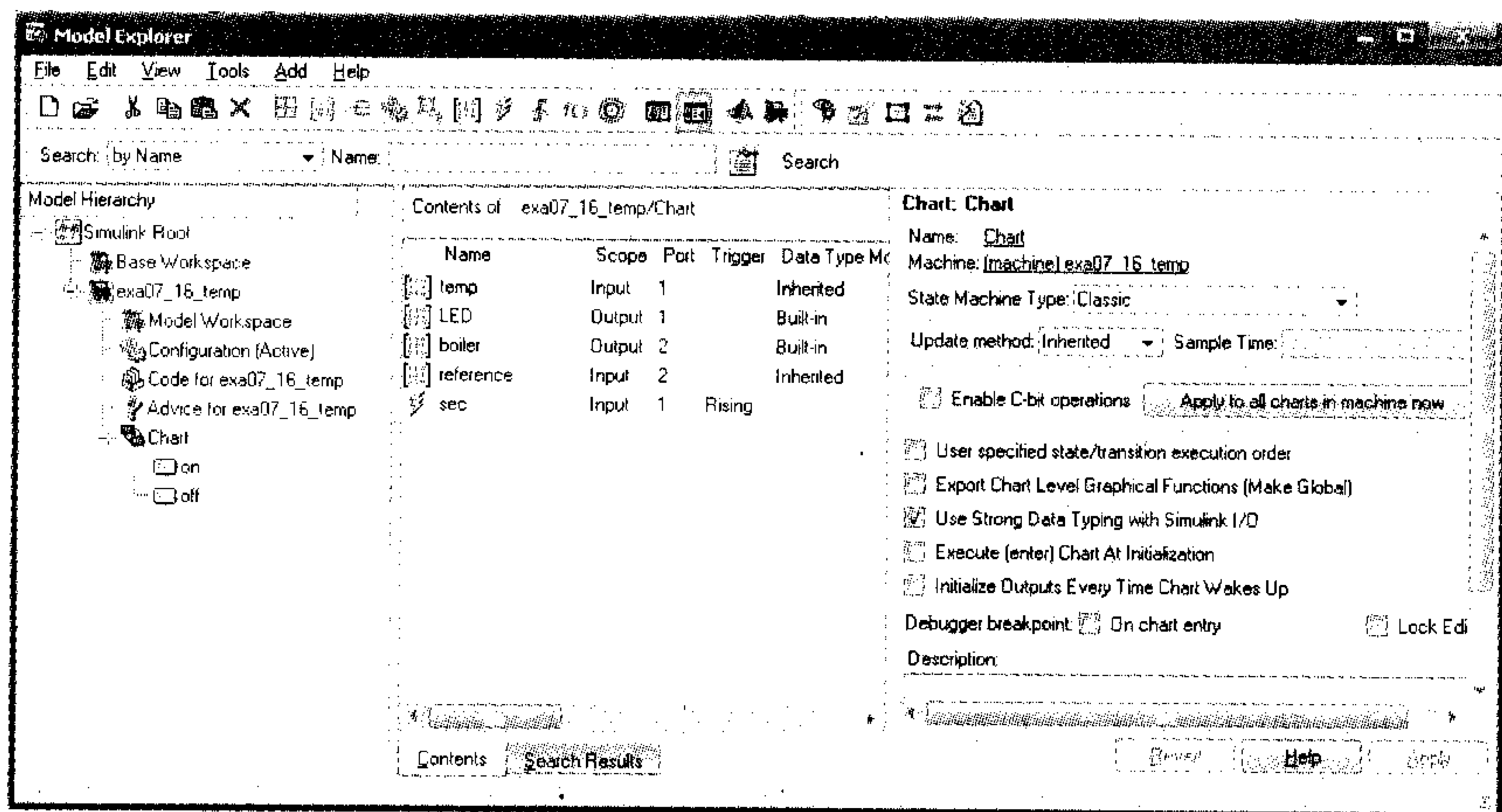


图 6-111　模型浏览器

8. 目标代码的生成

完成建模后，用户就可以使用 Stateflow 代码生成器，生成模型中控制部分的代码，来充当控制器。在代码生成过程中，Stateflow 可以产生两种类型的目标 ANSI-C 代码，即 Simulation Targets（仿真目标）和 Real-Time Workshop Fargets（Real-Time Workshop 目标）。Stateflow 默认产生 Simulation Targets 目标代码，函数名称为 sfun。

1）仿真目标代码

为了支持 Stateflow 符号图模型仿真，Stateflow 模型在编译时将自动产生一个 S-函数（MEX-文件）。当然，在代码产生时，可以不使用默认目标函数名称 sfun，而通过执行【Add】→【Target】命令来设置所需的函数名称和相应的参数，如图 6-112 所示，我们设置目标函数名称为 sf_boiler。

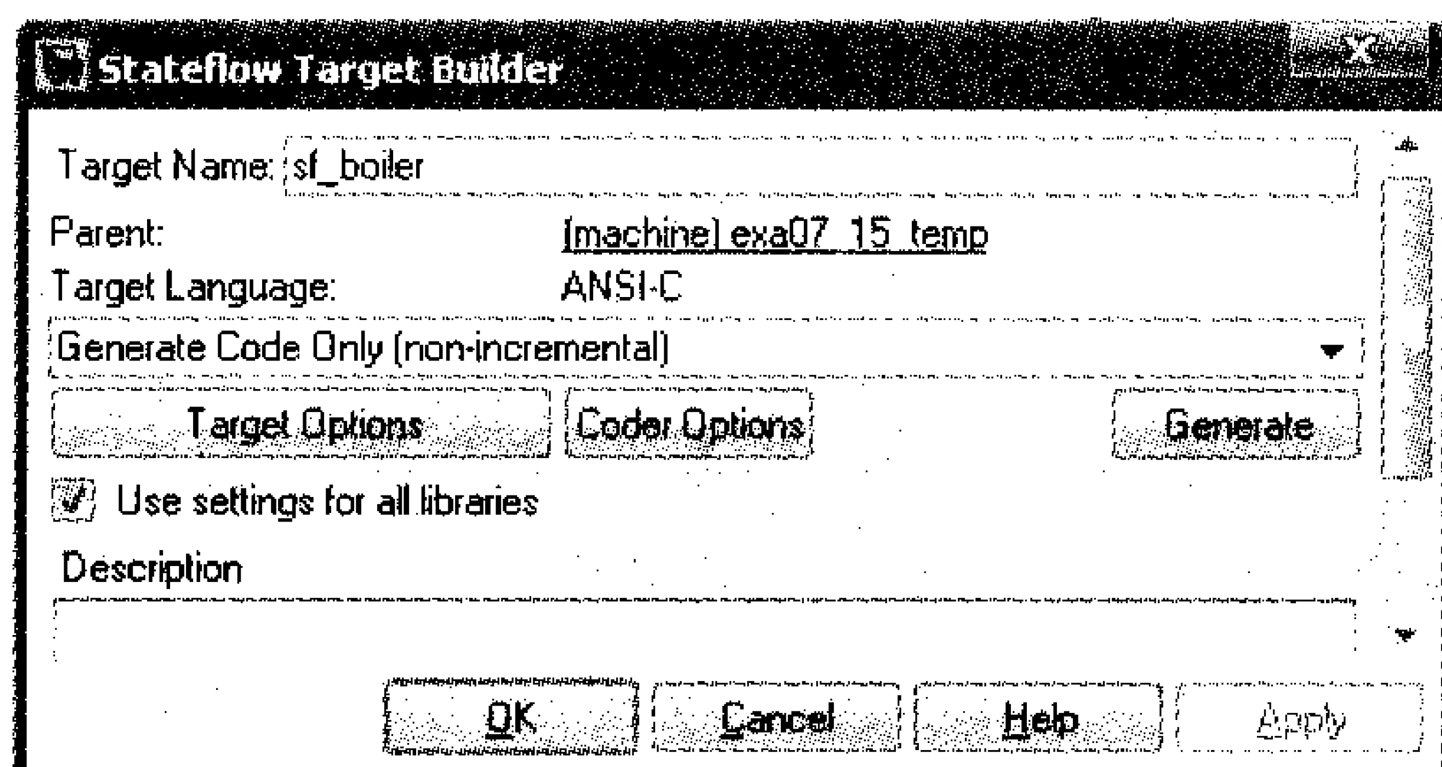


图 6-112　目标函数名称及其参数设置

如果需要更改目标代码函数的参数，可以通过执行【Tools】→【Open Simulation Target】命令来实现。

2）Real-Time Workshop 目标代码

Stateflow 还可以通过代码产生器将生成的目标代码无缝地嵌入到 Real-Time Workshop 产生的实时代码中。在 Stateflow 编辑窗口中可以通过执行【Tools】→【Open RTW Target】命令来打开 Real-Time Workshop 参数设置对话框。

6.6.3　Stateflow 常用命令

Stateflow 常用的命令有以下几个。

- stateflow 命令启动 Stateflow 编辑环境，绘制 Stateflow 流图。
- sfnew 命令创建一个新的带有 Stateflow Chart 模块的 Simulink 模型。如果 sfnew 命令后跟模型名称，则在建立新模型时将同时给出模型的名称。
- sfexit 命令关闭所有包含 Stateflow 模型的窗口，并退出 Stateflow 环境。
- sfsave 命令保存用户绘制的 Stateflow 模型。
- sfprint 命令打印绘制的 Stateflow 模型。

6.6.4　Stateflow 建模方法及实例

随着系统的复杂化，在进行系统仿真时使用 Stateflow 的情况越来越多。Stateflow 能够

实现有限状态机系统的仿真，图形化的建模环境使仿真的实现十分方便。下面通过一个的实例介绍 Stateflow 的控制系统建模与应用。

例 6.15 用 Stateflow 技术建立锅炉的控制系统模型。

首先通过 Simulink 构建锅炉子系统模型 Boiler Plant Model，如图 6-113 所示。锅炉模型的具体实现过程这里不做详细的讨论。

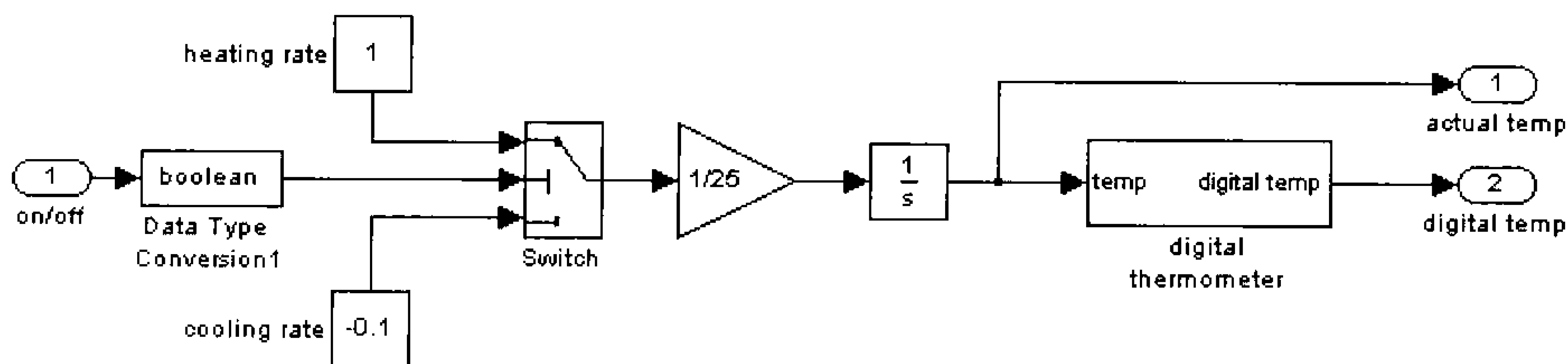


图 6-113　Boiler Plant Model 锅炉子系统模型

现在我们利用 Stateflow 技术实现锅炉的控制系统。该系统通过参考温度，时间触发信号，相应的图形控制逻辑，以及输出指示灯亮数据和开关量数据来控制锅炉的开启与关闭。程序初始化时，通过 turn_boiler 图形函数关闭锅炉。同样利用该函数，使锅炉 LED 指示灯每 5 秒钟闪烁一次。关闭 40 秒后，如果锅炉冷却，锅炉将被开启。加热 20 秒后，锅炉将再次关闭。重复以上循环。

在锅炉温度的调节过程中，可以通过 Simulink 示波器来观察锅炉温度的情况。LED 指示灯的闪烁情况如下，当锅炉关闭时灯熄灭，烧水过程亮红灯，水沸腾时亮绿灯。为实现锅炉温度控制功能，按以下步骤来建立 Stateflow 控制模型。

（1）进入 Chart 编辑环境。通过 Stateflow 放置 Chart 模型，并命名为 Bang-Bang Controller，具体实现过程请参考前文的“Chart 模型的编辑环境”相关内容。

（2）放置 box 模块。在 Chart 编辑窗口中放置一个 box 模块，并设置其标签为 Heater，如图 6-114 所示。

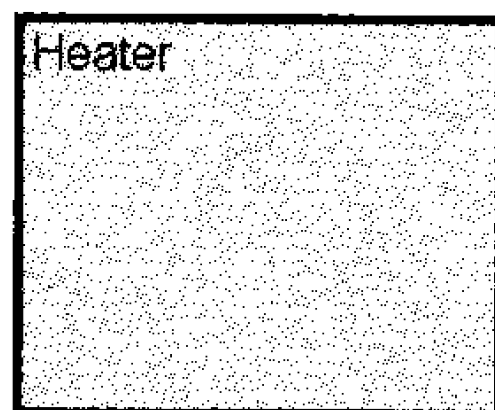


图 6-114　Heater box 模块

（3）添加状态模块。在 Heater box 中放置两个状态模块，标签分别为

```
Off
entry: turn_boiler(OFF)
on every(5,sec): flash_LED()
```

和

```
On
en: turn_boiler(ON)
du: flash_LED()
```

同时将第二个模块设置成 Subcharted。方法是选中 On 状态模块，单击鼠标右键，在弹

出的快捷菜单中执行【Make Contents】→【Subcharted】指令，结果如图 6-115 所示。

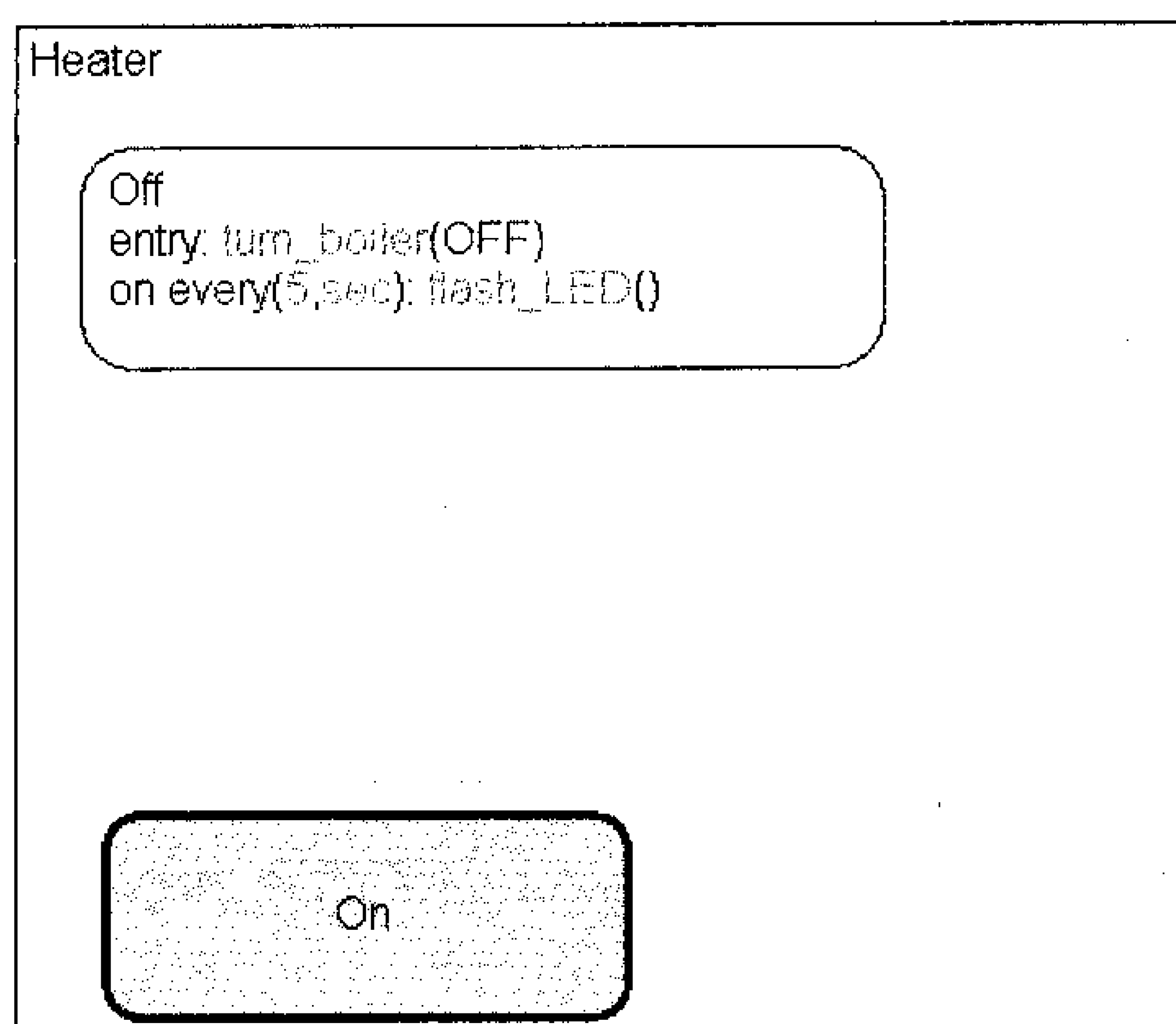


图 6-115　Off 和 On 状态模块

（4）建立 Off 和 On 状态模块的迁移。这里先建立 3 个迁移状态，一个为默认状态，指向 Off 状态模块；另一个为 Off 状态迁移到 On 状态，在该迁移中当 Off 状态激活 40 秒后，如果锅炉温度低于参考温度（20℃），则 Off 状态的激活迁移到 On 状态；还有一个为 On 状态迁移到 Off 状态，在该迁移中，当 On 状态激活 20 秒后，On 状态的激活迁移到 Off 状态。模型如图 6-116 所示。

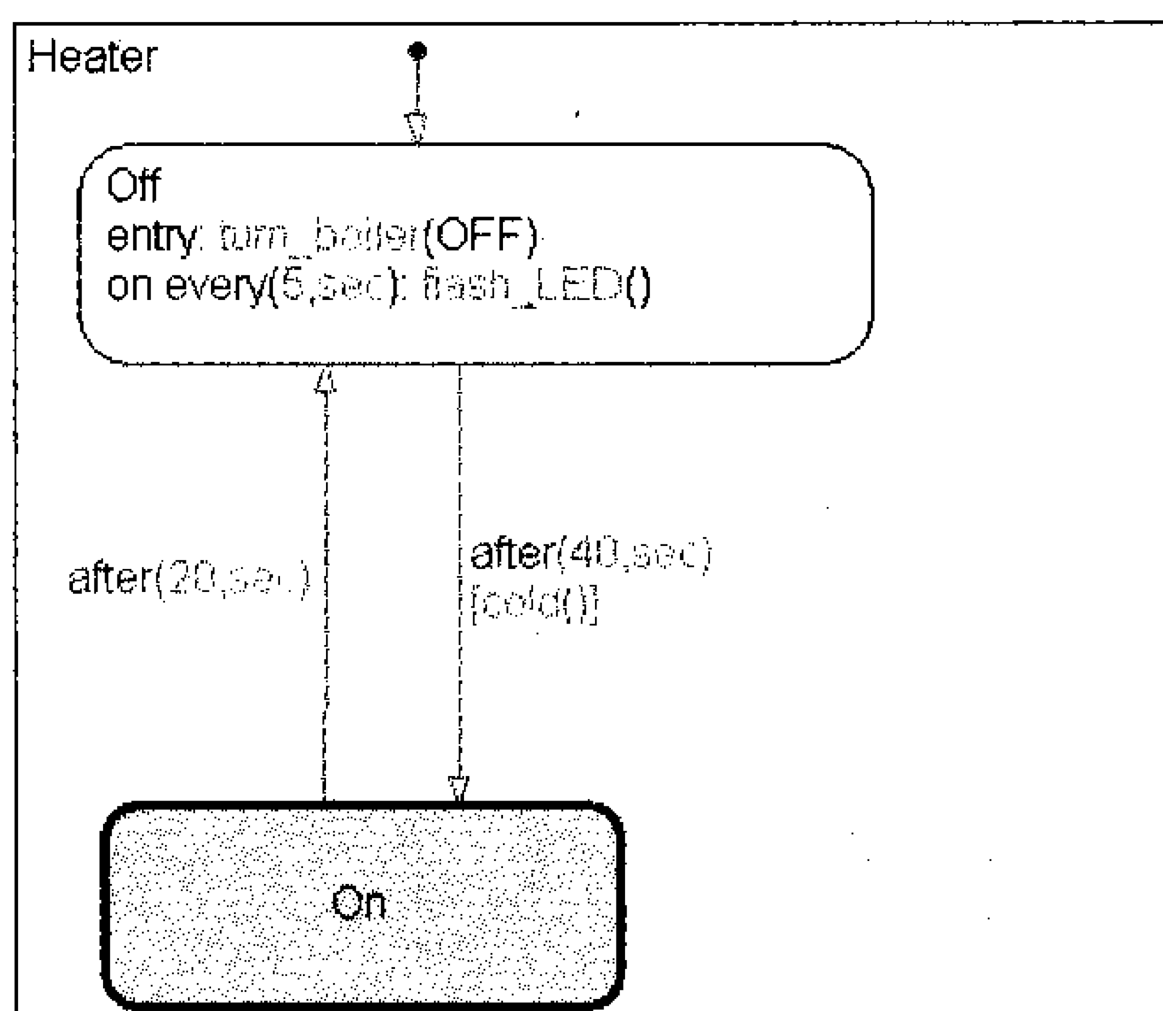


图 6-116　Off 和 On 状态的迁移设置

（5）通过 Stateflow chart 编辑窗口执行【Add】→【Event】命令，增加从 Simulink 输入事件“sec”，选用上升沿触发模式，通过执行【Data】命令，增加“reference”和“temp”两个从 Simulink 输入的固定点数据变量，增加“LED”和“boiler”两个输出到 Simulink 的数据变量，增加“Green”、“OFF”、“ON”和“RED”4 个常值变量，增加“color”局部变量。

（6）绘制 turn_boiler、flash_LED 和 cold 图形函数。这些函数的设置主要采用“Default Transition”、“Connective Junction”和“Function”3 种工具。操作方法是在刚建立的 Chart 模型中，通过“Function”工具放置标签分别为 turn_boiler(mode)、flash_LED()和 cold()的 3 个函数，然后分别双击这 3 个函数方框，进入函数编辑窗口后，利用“Default Transition”和“Connective Junction”工具建立如图 6-117、图 6-118 和图 6-119 所示的逻辑图形函数。

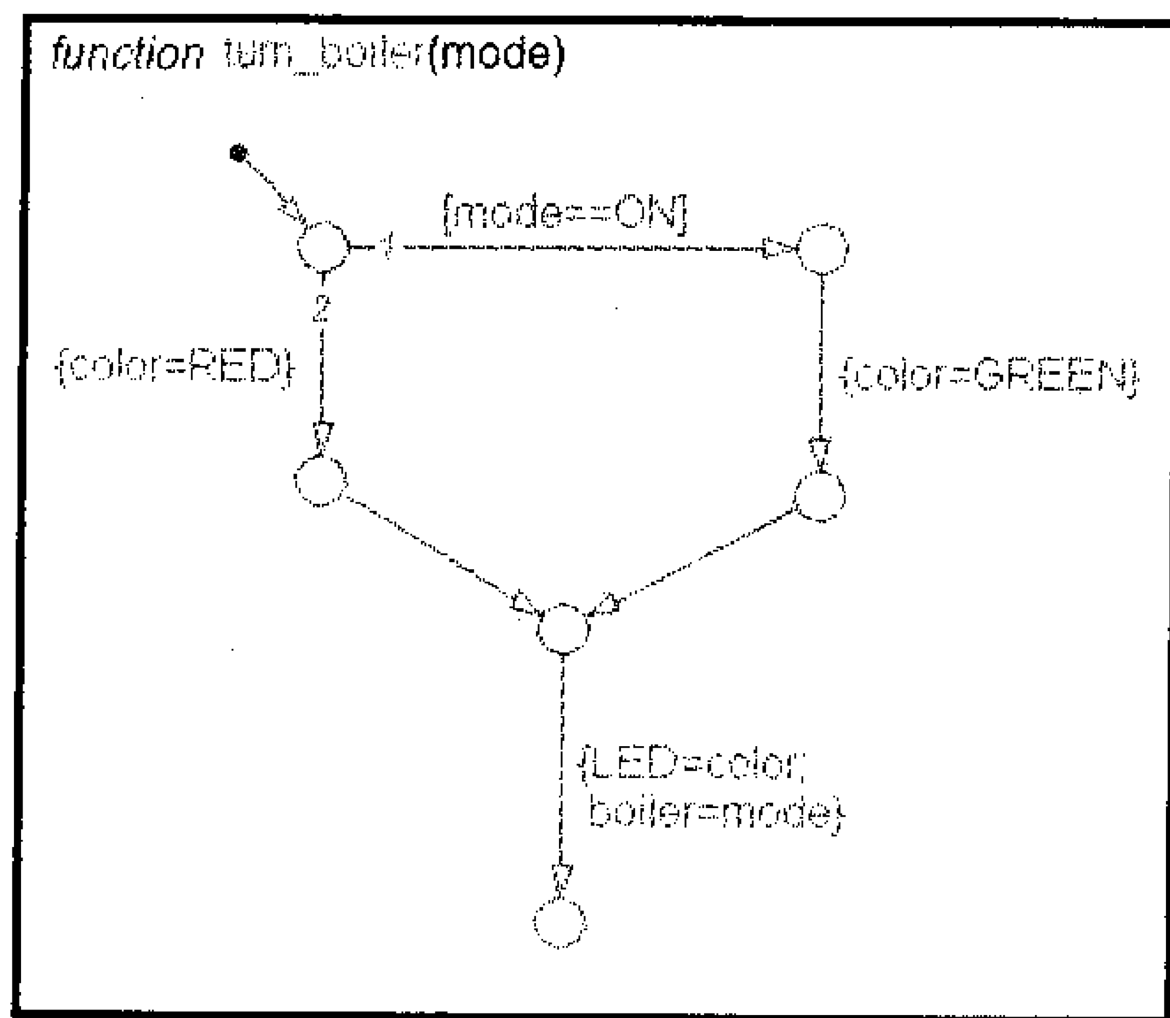


图 6-117　turn_boiler 图形函数内容

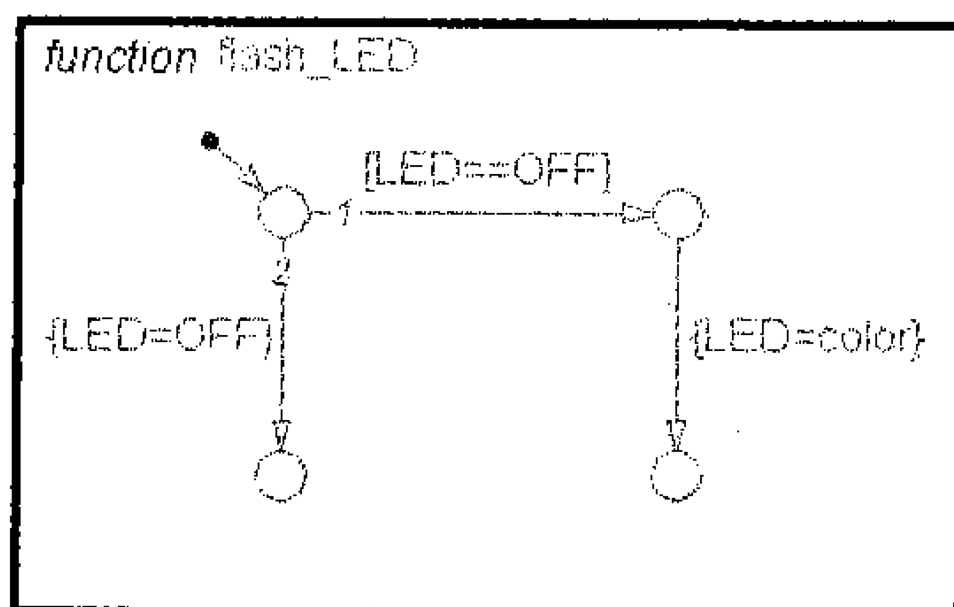


图 6-118　flash_LED 图形函数内容

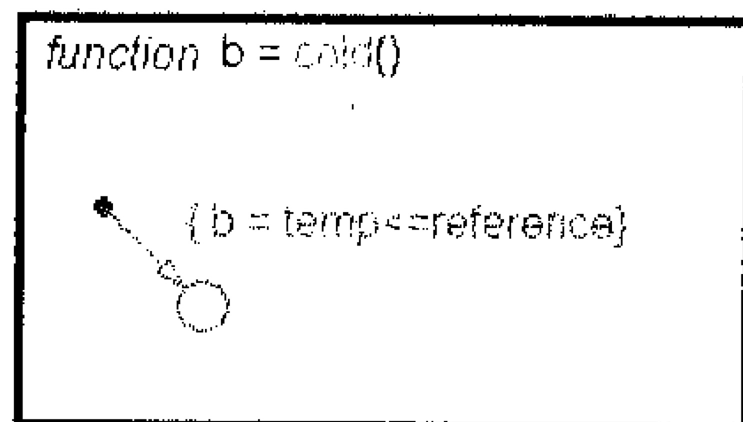


图 6-119　cold 图形函数内容

（7）通过 subcharted 方式，建立 On 状态的模块的迁移模式。双击 On 状态模块，进入状态编辑窗口，通过“History Junction”、“Default Transition”和“Connective Junction”3 个工具，建立的 On 状态模块的迁移模式如图 6-120 所示。图中包括 HIGH 和 NORM 两个状态模块、一个 warm 图形函数，以及 3 条迁移连线和一个历史交汇。由于 On 状态模块采用了 subcharted 方式，此时可以设置超级迁移端口，图中的 Heater On warm()迁移就采用了这种方式。

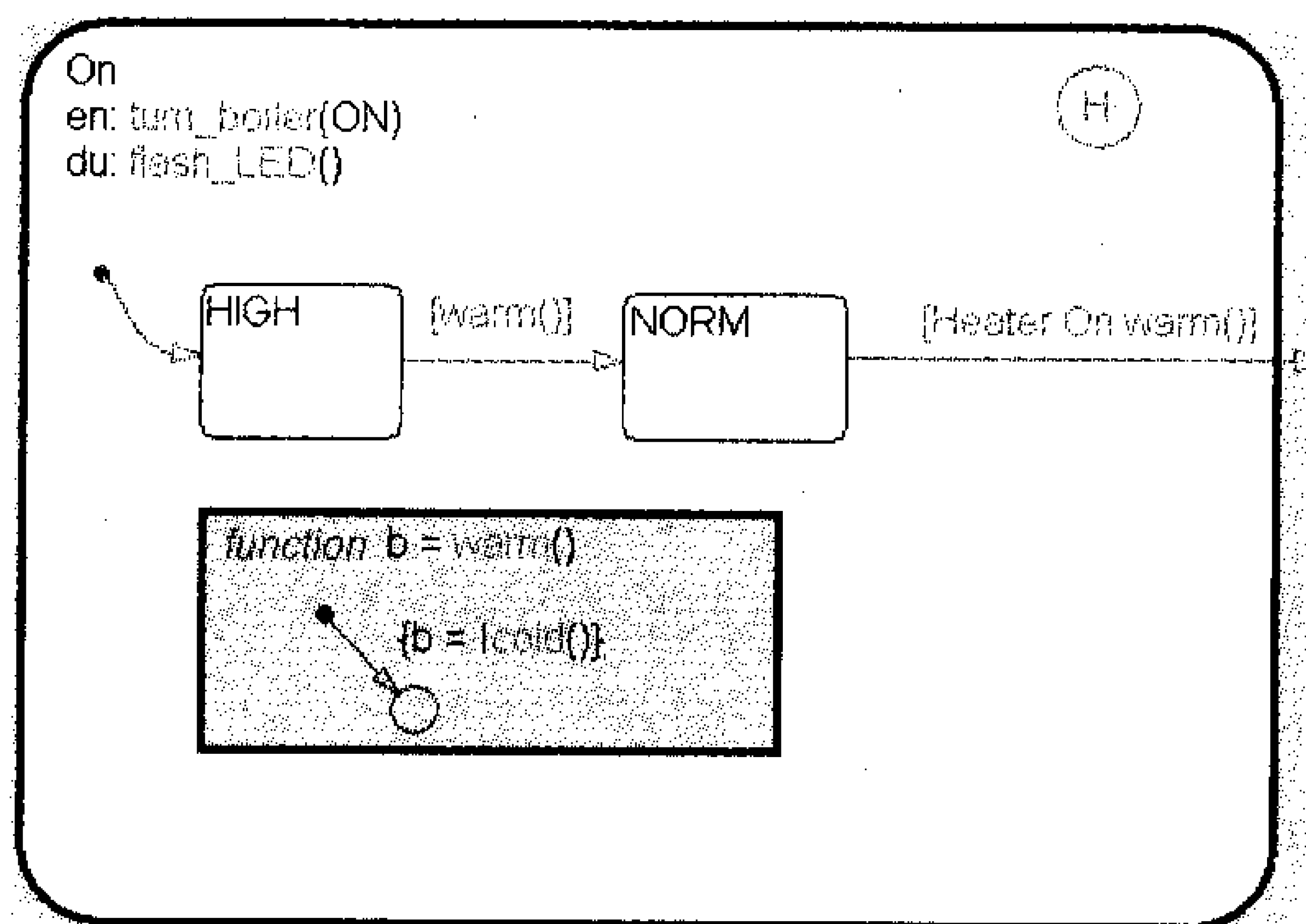


图 6-120　On 状态的迁移模式

（8）从 On 状态模块的超级迁移端口，通过 warm()图形函数，迁移 On 状态到 Off 状态。至此，构建的锅炉控制系统模型如图 6-121 所示。

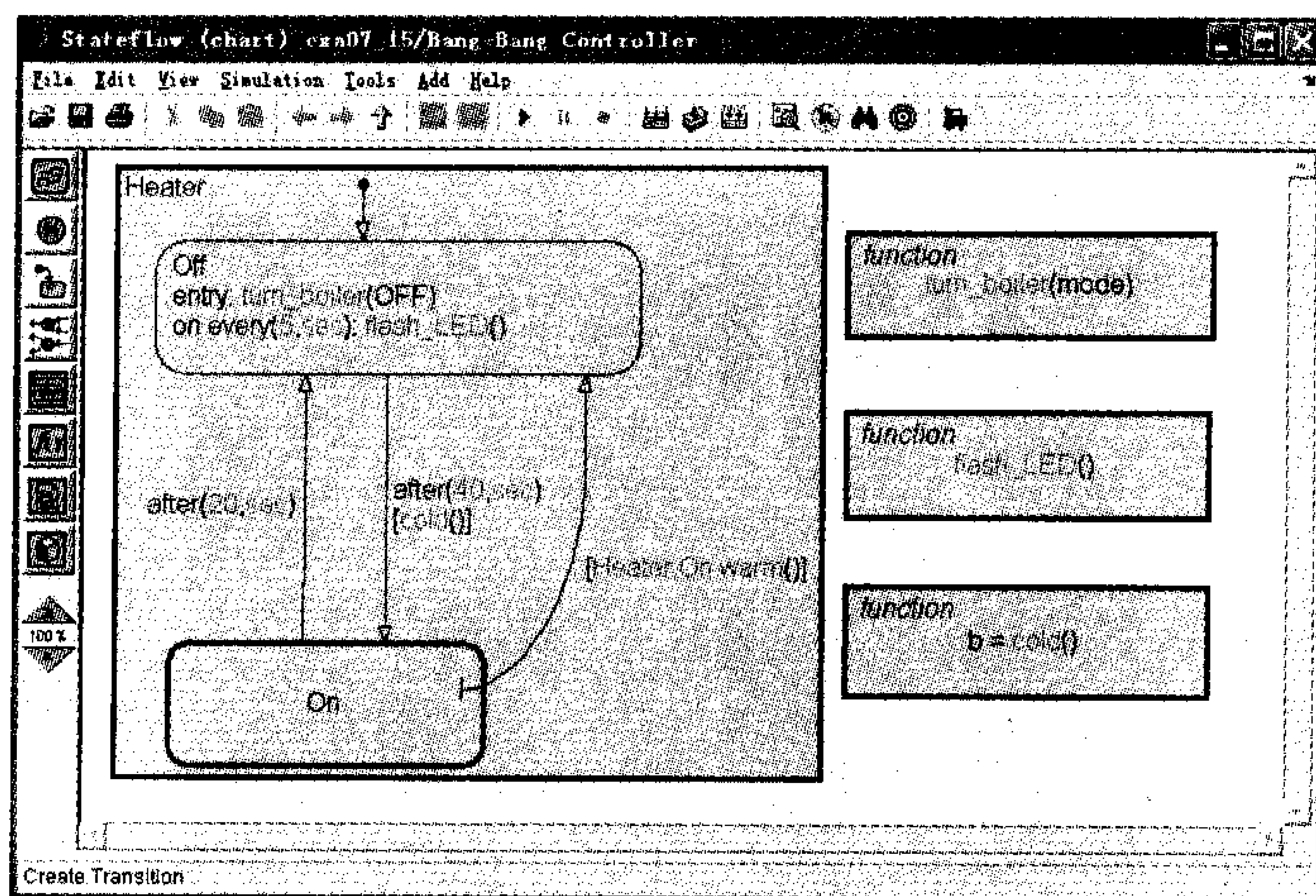


图 6-121　Bang-Bang Controller 锅炉控制系统 Stateflow 模型

（9）在锅炉模型的 Simulink 窗口，增加一个“Pulse Generator”、一个“Constant”和一个“Scope”模块，其中“Pulse Generator”的参数设置如图 6-122 所示。连接这 3 个模块，以及“Bang-Bang Controller”和“Boiler Plant Model”模块。最终构建的 Stateflow 控制的锅炉模型系统如图 6-123 所示。

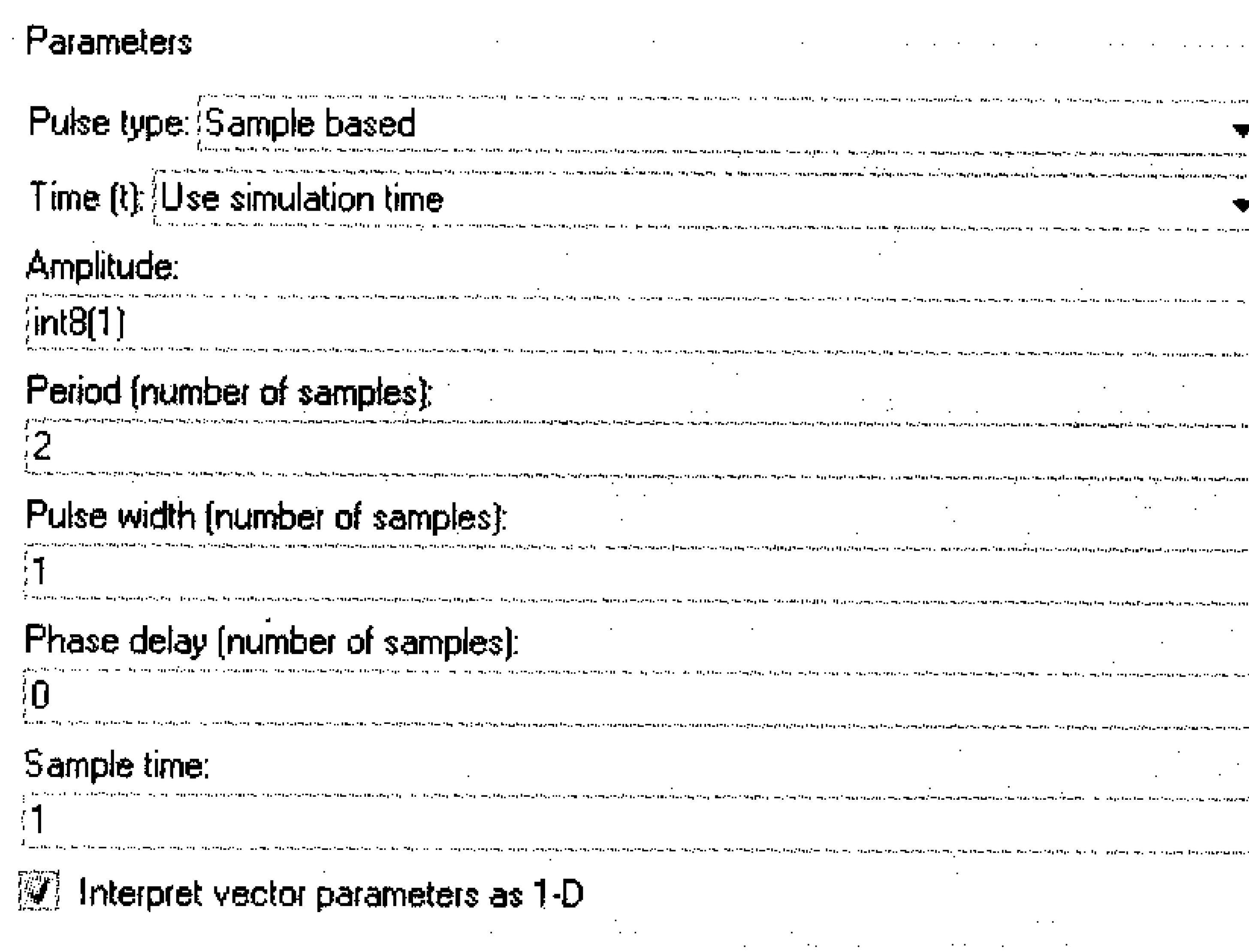


图 6-122　Pulse Generator 参数设置

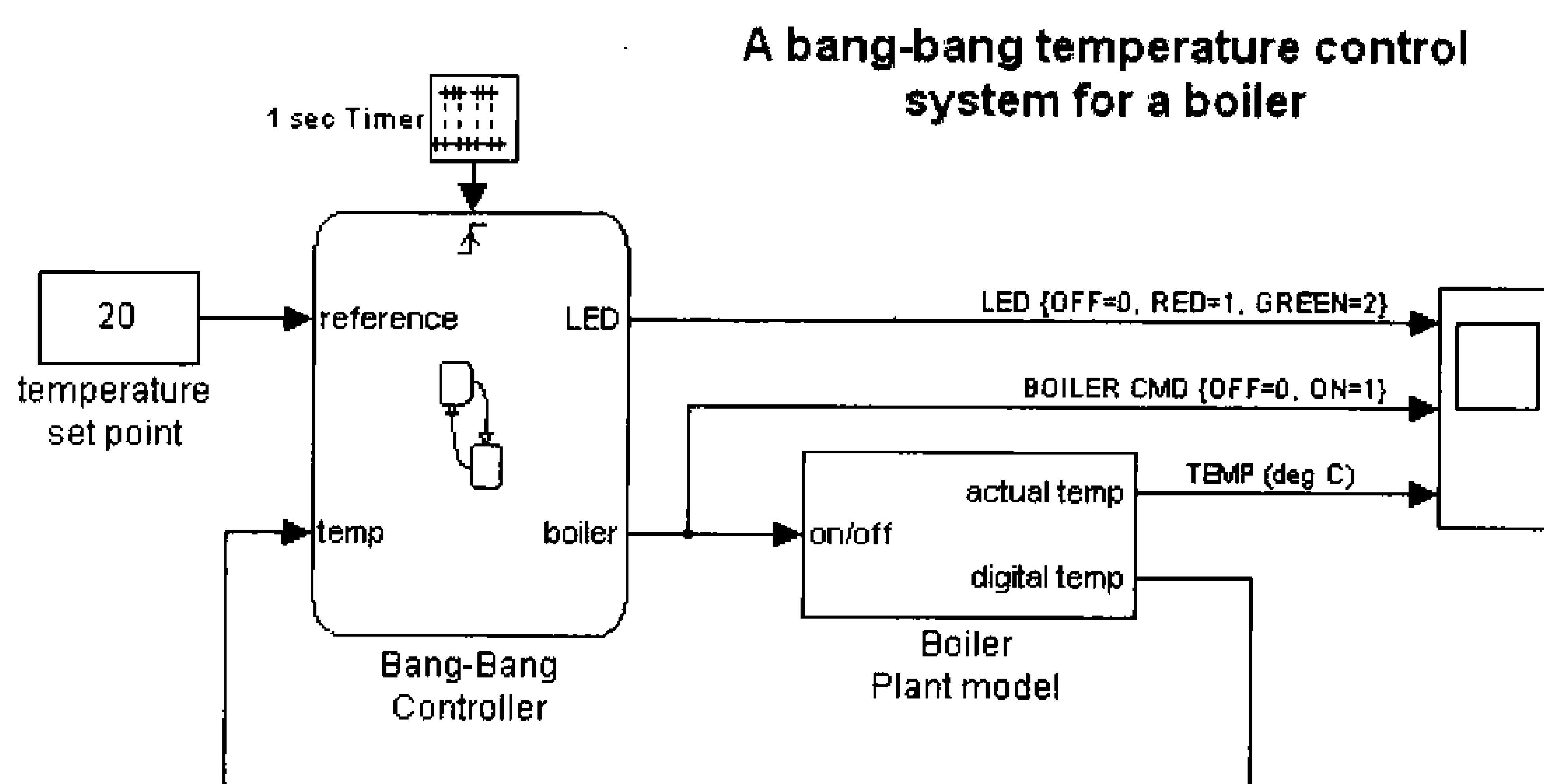


图 6-123　Stateflow 控制的锅炉模型系统

（10）运行图 6-123 所示的锅炉模型系统，其结果如图 6-124 所示。仿真结束后，图 6-123 的锅炉模型系统会变成彩色。

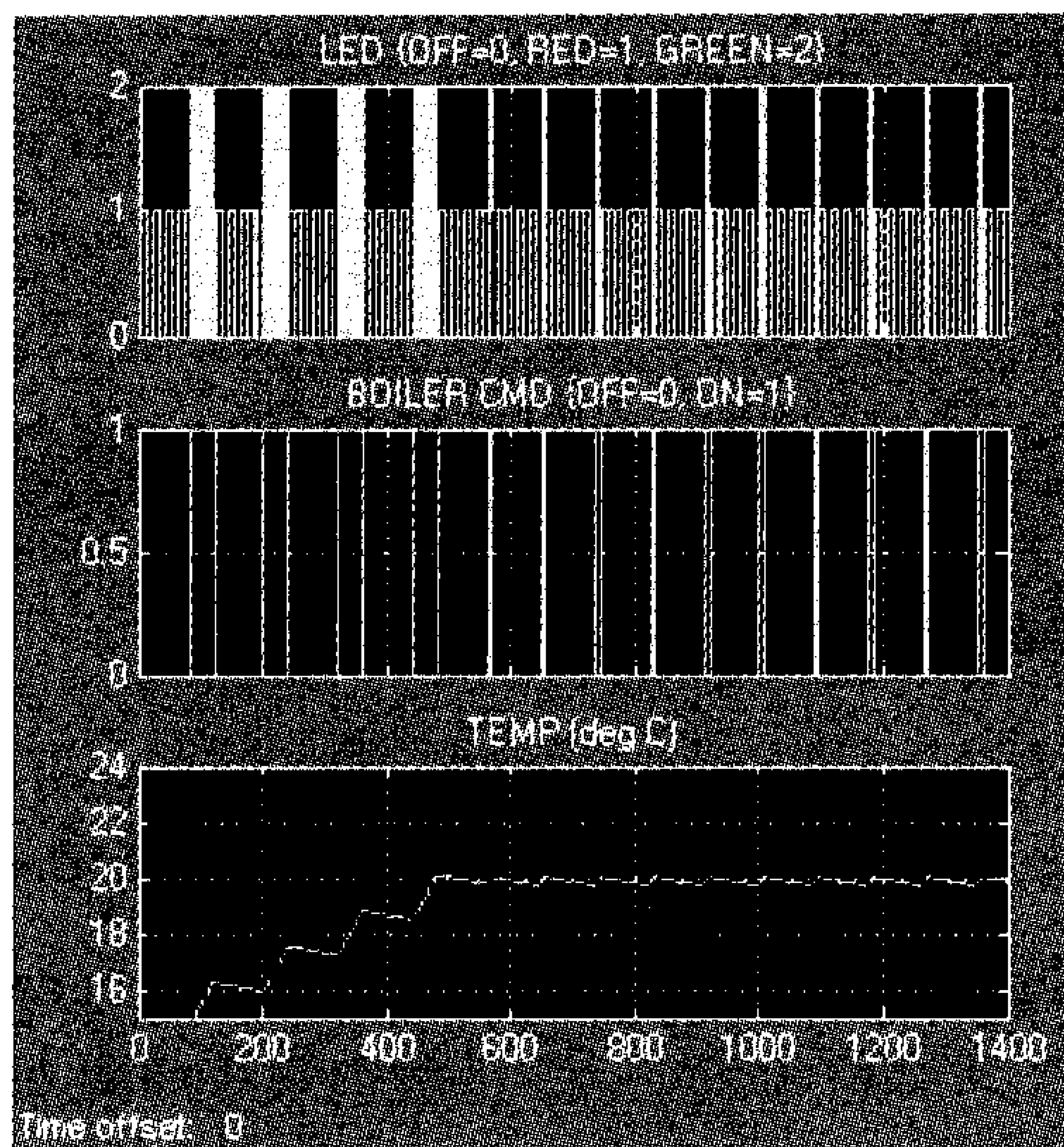


图 6-124 锅炉模型系统的仿真结果

6.7 Simulink 模型的实时代码生成技术

通过 Real-Time Workshop（RTW），可以生成 Simulink 模型的实时代码。本节在介绍 RTW 功能和组成等基本知识的基础上，通过实例，阐述了 RTW 生成 Simulink 普通实时程序和实时代码的方法。

6.7.1 Real-Time Workshop 介绍

RTW 是 Simulink 和 MATLAB 功能的一种拓展，它提供了一个快速生成代码的环境，可以自动编译 Simulink 模型，生成实时代码。

1. 组成与功能

RTW 是和 MATLAB、Simulink 一起使用的工具，一方面它可以直接从 Simulink 模型生成代码，并且可以自动建立在不同环境下运行的程序，这些环境包括实时系统和单机仿真；另一方面，通过 RTW，用户还可以在远程处理器实时的运行 Simulink 模型，并且它所生成的单机仿真程序同样具备在外部计算机运行的能力。

RTW 的主要组成及其特性如下。

- Simulink Code Generator：自动为 Simulink 模型产生 C 代码。
- Make Process：Real-Time Workshop 的拓展功能，可以让用户通过自定义编译和连接，来达到应用目的。

- Simulink External Mode：外部模式充当 Simulink 与实时测试环境之间的信息桥梁。外部模式在使用 Simulink 时，作为前台终端进行实时参数的凋整，数据记录和可视化。
- Targeting Support：将目标与 Real-Time Workshop 捆绑在一起，就可以建立一个实时或者模型环境。一般的实时环境和其他绑定的目标为自定义快速建模或者目标产品提供了一个框架。
- Rapid Simulations：使用 Simulink Accelerator，S-Function Target，或者 Rapid Simulation Target，可以将仿真速度提高 5～20 倍。利用 Simulink Accelerator，S-Function Target，或者 Rapid Simulation Target 产生代码是经过高度优化的，为特定模型设置的算法。
- Large-Scale Modeling：支持 Simulink 中的多级建模映射到 Real-Time Workshop，从而可以产生多个独立的模型。

RTW 能够应用的场合也十分广泛，主要包括以下几个。

- 实时控制——读者可以使用 MATLAB 和 Simulink 设计控制系统，并且从建立的模块图表模型生成代码。读者可以编译和载入它们到目标硬件。
- 实时信号处理——读者可以使用 MATLAB 和 Simulink 设计信号处理算法，同样可以从模型生成代码，编译和载入它们到目标硬件。
- 硬件环路仿真——读者可以建立模仿实际工程测量，系统动力和激励信号的 Simulink 模型。从模型生成的代码可以被定位到特殊用途的硬件（如 DSP），它提供了物理系统的实时表示。
- 交互的实时参数调整——读者使用 Simulink 作为建立的实时程序的前端，就可以利用 Simulink 的图形界面，在程序执行时改变参数。
- 高速的单机仿真。
- 生成可插入其他仿真程序的便携 C 代码。

无论是哪种应用，代码生成都是其中必需的过程。RTW 生成的代码在默认情况下是高度优化和完全注释的 C 代码。从任何 Simulink 模型都可以生成代码，它们包括：线性和非线性模型系统，以及连续、离散以及混合模型系统。

所有的 Simulink 模块都自动地转化为代码，除了 MATLAB 的函数模块和调用 M 文件 S-函数模块。如果想把它们和 RTW 一起使用，就需要将这些模块重新改写为 C MEX S-函数。

RTW 包含一系列的目标文件，并由目标语言编译器（TLC）编译生成 ANSI-C 代码。目标文件是 ASCII 文本文件，描述如何将 Simulink 模型转化为代码。对于高级用户，目标文件使得它能够自定义生成代码。

读者还可以把 C MEX S-函数和生成的代码一起合并到可执行的程序，同样还可以为自己的 C MEX S-函数编写一个目标文件来内嵌 S-函数，这样就可以通过减少对 S-函数的调用来提高性能。

RTW 可输出的代码类型有以下几种：

- C 代码——为 Simulink 模型生成包含系统方程和初始化函数的代码，读者可以在非实时环境或者实时环境使用这些代码。

- Ada代码——从Simulink模型生成Ada代码，这要求用户安装Real-Time Workshop Ada Coder。
- 实时程序——将代码转换为适合指定硬件实时运行的程序。对应的代码被设计为和一个外部时钟源相连接，并且以一个固定的由用户设置的采样速率运行。
- 高性能的单机仿真——将生成代码和普通实时系统目标文件一起使用，为单机仿真生成可执行程序。在仿真结束时，可执行文件将产生一个model.mat文件，它里面包含了Simulink保存仿真数据的MATLAB变量，这个文件可以在MATLAB中进行分析。

2．基本概念

在学习和使用RTW时，读者必须了解RTW中的一些基本概念，主要包括：普通实时，目标定位，目标语言编译文件，建立过程，模板makefile，配置模扳文件，以及设置模型参数。

1）普通实时

RTW提供了普通的实时发展对象，所谓普通实时是一个在单任务或者多任务模式下仿真固定步长的模型环境，也是实现代码验证的一种方法。

2）目标定位

使用RTW，读者必须选择放置生成代码运行的环境，包括硬件或者操作系统，这称为目标定位，环境本身就是目标（Target)。而宿主（Host）就是用户运行MATLAB、Simulink和RTW的系统。使用RTW，可以生成代码以及能够在目标系统中的可执行程序，生成特定目标的代码的过程由系统目标文件、模板makefile文件和make命令来控制。系统目标文件和模板makefile文件定义了目标的类型，它们必须在仿真参数对话框的Real-Time Workshop页中设置。

3）目标语言编译文件

目标语言编译文件（Target Language Compiler files或TLC)，是供目标编译器编译和执行的文件。它描述了如何将Simulink模型翻译为用户所设置的目标代码。RTW使用TLC文件来把Simulink模型翻译成代码，而系统目标文件是TLC程序生成可执行程序的切入点。

4）建立过程

RTW建立过程由make_rtw来控制，这个程序在用户单击Real-Time Workshop页上的【Build】按钮后被调用。首先，make_rtw编译模块图表，并生成一个model.rtw文件。其次，make_rtw调用目标语言编译器来生成代码，但系统目标文件必须事先设置。再次，make_rtw从Real-Time Workshop页上指定的模板makefile生成一个makefiel，名为model.mk。最后，如果运行的host和模板makefile的HOST宏匹配，那么就调用make程序，生成代码建立程序。

5）模板makefiles

RTW使用模板makefile从生成代码建立可执行程序。一般而言，模板makefile文件名与应用目标相对应，其后缀名为.tmf。例如，grt_unix.tmf是用于UNIX的普通实时模板makefile，这个模板文件应用的目标是gr_unix。

从模板makefile生成的makefile逐行地复制模板makefile的每一行，并且把记号扩展

到 maekfile。生成的 makefile 名为 model.mk，它被传给 make 命令，make 再根据此文件来生成可执行程序。

6）模板 makefile 配置

读者可以通过修改模板 makefile 来配置建立过程。完成的过程可以是，把模板文件复制到自己的当前工作目录，并编辑它。或者可以通过在 make_rtw 命令后面添加选项来配置模板 makefile。例如，为 grt_unix.tmf 打开调试符，读者可以在仿真参数对话框设置相应的建立命令：

```
make rtw OPT_OPTS=-g        %OPT_OPTS=-g 就是添加的选项设置
```

7）设置仿真参数

修改仿真参数可以控制仿真的行为，如起始和终止时间。仿真参数直接影响到代码生成和程序建立，所以在生成代码和建立程序之前，读者必须检验仿真参数设置的正确性。关于仿真参数的设置，前面章节已经介绍得很清楚，这里就不再赘述。

3. 第三方编译器选择

大多数目标创建一个可在工作站上执行的目标。在创建可执行目标之前，RTW 必须和适当的编译器进行连接。在 Windows 中，RTW 可以使用的编译器版本如表 6-19 所示。

表 6-19 Real-Time Workshop 可以使用的编译器

编 译 器	版 本	编 译 器	版 本
Borland	5.0，5.5，6.0	Intel	7.1
LCC	MATLAB 自带的 LCC	Microsoft Visual C/C++	5.0，6.0，7.0，8.0

下向就介绍如何配置系统，使得 RTW 能够使用编译器。

1）Borland

确保 Borland 环境变量已经被定义，正确地指定 Borland 编译器所在的目录。是否定义了这个 Borland 环境变量，可以在 DOS 窗口输入：

```
Set BORLAND
```

如果 Borland 环境变量已经定义了，那么就会返回编译器所在的目录。如果 Borland 环境变量没有定义，则必须指定安装 Borland 编译器所在的位置。

如果系统为 Windows 95 或 Windows 98，那么就将

```
Set BORLAND=<编译器的路径>
```

加入到 autoexec.bat 文件中。

如果系统为 Windows NT、Windows 2000 或 Windows XP，在系统控制面板中，单击“高级”选项卡，选择“环境”选项，然后将 Borland 的路径添加进去。

2）Intel

RTW 同样支持 Intel 编译器（Version 7.1 应用于 Microsoft Windows），Intel 编译器需要 Microsoft Visual C/C++6.0 或更新的版本。然而，只有 Visual C/C++6.0 能够和带有 Intel 编译器的 RTW 联合工作。

在 RTW 生成代码时为了使用 Intel 编译器，要用 mex-setup，然后选择 Intel 编译器。

手动设置 Intel 编译器的操作步骤如下。

（1）复制文件$(MATLAB)\bin\win32\mexopts\intelc71opts.bat 到期望的目录中，并命

名为 mexopts.bat。

（2）编辑新文件 mexopts.bat，用 Microsoft Visual C/C++ Version 6.0 安装根目录替换%MSVCDir%。

（3）用 Intel Compiler 安装根目录替换%INTELC71%。

3）LCC

LCC 是自由软件，是 MATLAB 的默认安装。只要利用 MATLAB 自带 LCC 就能和 RTW 共同工作。

4）Microsofl Visual C/C++

利用 MATLAB 命令

```
mex -setup
```

来定义 Visual C/C++ Versions 5，6，7 和 8 的使用环境。

在 Windows 系统中，S-函数目标和模型参考仿真目标都是借助 MATLAB 的 mex 命令产生 DLL 文件的。

5）Watcom

Watcom C 编译器已经由 OpenWatcom 组织接管（http://www.openwatcom.org），RTW 目前还自带了 Watcom 相关的目标配置，但是这一举措也许会在今后发生改变。

确保 Watcom 环境变量已经被定义，正确地指定 Watcom 编译器所在的目录，可以在 DOS 窗口输入命令：

```
set WATCOM
```

如果 WATCOM 环境变量已经定义了，那么就会返回编译器所在的目录；如果 WATCOM 环境变量没有定义，必须指定安装 WATCOM 编译器所在的位置。如果系统为 Windows 95 或 Windows 98，那么就将

```
set Watcom=<编译器的路径>
```

加入到 autoexec.bat 文件中。

如果系统为 Windows NT、Windows 2000 或 Windows XP，在系统控制面板中，单击“高级”选项卡，选择“环境”选项，然后将 Watcom 的路径添加进去。

如果在设置过程中，遇到“Out-of-Environment Error Message”错误信息，可以用鼠标右键产生这个问题的程序（如 autoexec.bat），并且从弹出的菜单中选择【Properties】命令，从打开的对话框中单击“Memory”选项卡，并把 Initial Environment 属性设置为所允许的最大值，然后单击【Apply】按钮，应用以上设置。

6）编译器的优化设置

RTW 对每个支持的编译器都应用默认的优化设置，在极少数情况下，由于编译器的缺点，在 RTW 中应用编译器优化可能会导致生成的可执行程序产生错误的结果，即使代码本身是正确的。因此，用户通常要遇到如何设置编译器优化问题，例如降低优化设置或关闭优化设置。

6.7.2 Simulink 模型的普通实时程序生成方法与实例

从一个 Simulink 模型生成普通的实时程序有以下几个特点。

- 它作为一个单机程序执行，独立于外部的定时和事件。
- 它将数据保存在 MATLAB 的 MAT 文件中，用作后续数据分析。
- 它可以在 PC 或者 UNIX 环境下生成。

下面，我们将通过 Simulink 模型实例，讲述其普通实时程序的生成方法。

例 6.16 通过 f14 战斗机纵向运动控制模型，生成该模型的普通实时程序。

1. Simulink 模型

这里选用的模型为美国 f14 战斗机的纵向运动进行控制的仿真模型，该模型位于 $\MATLAB\R2007a\toolbox\Simulink\simdemos\aerospace 目录下，读者可以在 MATLAB 窗口中输入

```
>>f14
```

命令来打开 f14 模型，该模型如图 6-125 所示。

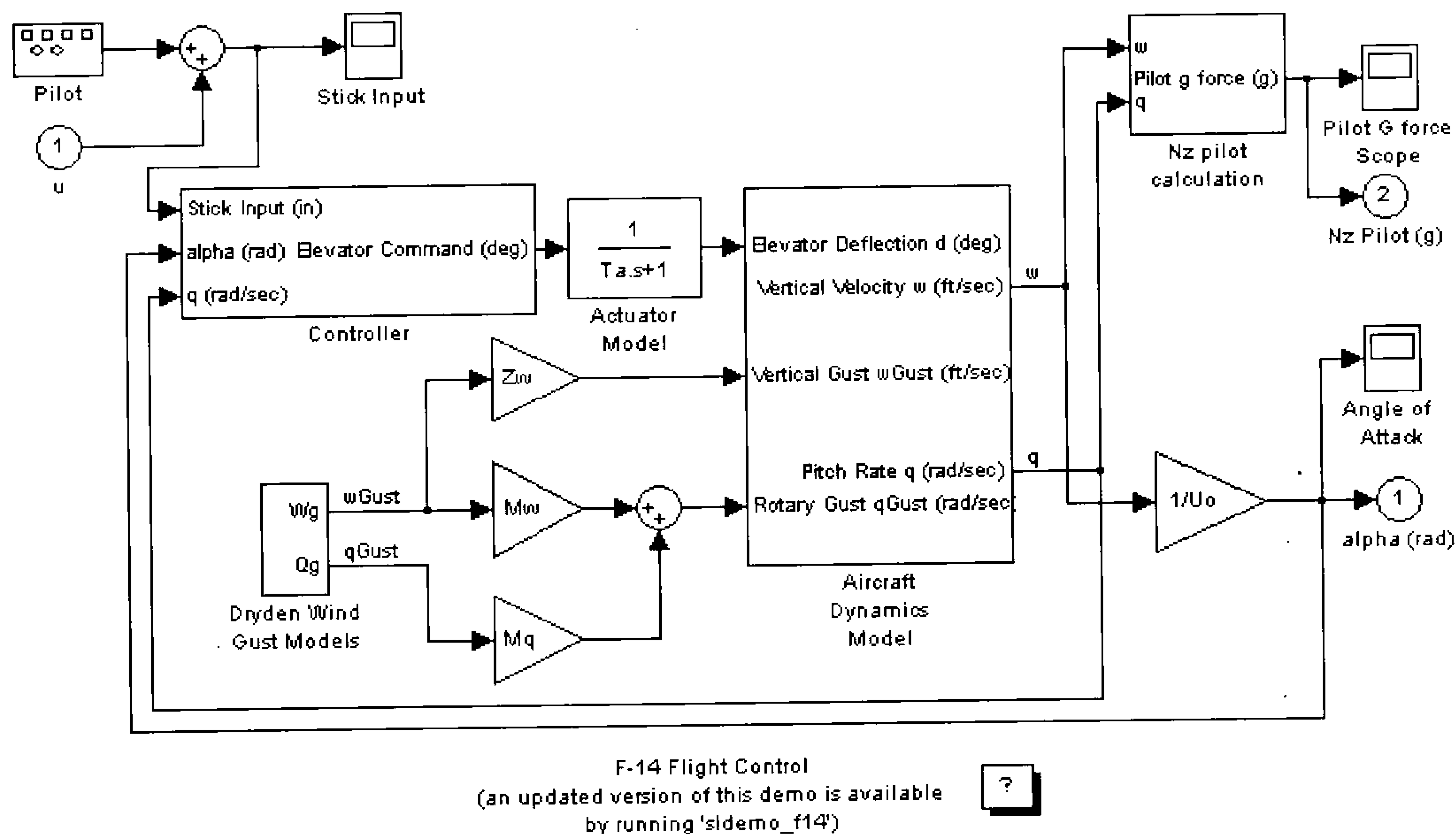


图 6-125 f14 战斗机纵向运动控制模型

据图 6-125 可知，f14 模型由 1 个信号发生模块、3 个示波器模块、4 个增益和 5 个子系统组成。其中，信号发生模块用来模拟飞行员的操作输入，它的波形由 Stick Input 示波器进行跟踪；Pilot G force Scope 用来显示飞行员所承受的重力；Angle of Attack 用于跟踪攻击的飞行角度；Controller 子系统模拟 f14 飞机的控制器；Actuator Model 子系统用于模拟 f14 飞机的驱动器；Dryden Wind Gust Models 子系统模拟 f14 飞机所受的风场力；Aircraft Dynamics Model 子系统实现 f14 飞机的动态系统模型的仿真；Nz pilot Calculation 子系统用于计算飞行员所受的重力；其他 4 个增益模块用于控制相应信号的增益。考虑到 Simulink 模型生成普通的实时代码与模型的内容及其复杂程度无关，因此各子系统的模型设计及其内部工作原理，感兴趣的读者可以通过打开子系统自行剖析。

2．实时代码的生成方法与步骤

模型建立之后，我们可以按照下面的方法与步骤来完成其普通实时代码的生成。

1）设置模型的参数

在生成实时代码之前，必须对程序的参数进行设置。在 f14 模型窗口中，通过执行【Simulatlon】→【Configration Parameters】命令打开仿真参数对话框，这个对话框的大部分页的参数设置内容在本章的前面内容已经详细讲过，但没有深入讲述 Real-Time Workshop 页参数的设置内容。

在设置 f14 模型的 Real-Time Workshop 页参数之前，首先设置模型的解算器（Solver）的参数，方法如下。

（1）在 solver 页，设置 Solver options Type 参数为 Fixed-step，并且选择 ode5（Dormand-Prince）解算器。

（2）设置 Fixed-Step Size 参数为 0.05。

设置完解算器的参数，我们再来设置 Real-Time Workshop 页的参数。Real-Time Workshop 页是用户设置模型的参数，选择模板制造文件（makefile），产生代码和建立程序的地方。对话框中的默认参数值不一定会与生成可实时执行代码所要求的参数匹配。因此，Simulink 允许用户在使用 RTW 时，可以改变默认参数值，以匹配模型的仿真参数。Real-Time Workshop 页的参数设置对话框如图 6-126 所示。如果选中 Generate code Only，则 RTW 只产生代码而不编译，意味着不生成目标文件和可执行文件。这里，我们选择系统默认的参数值。

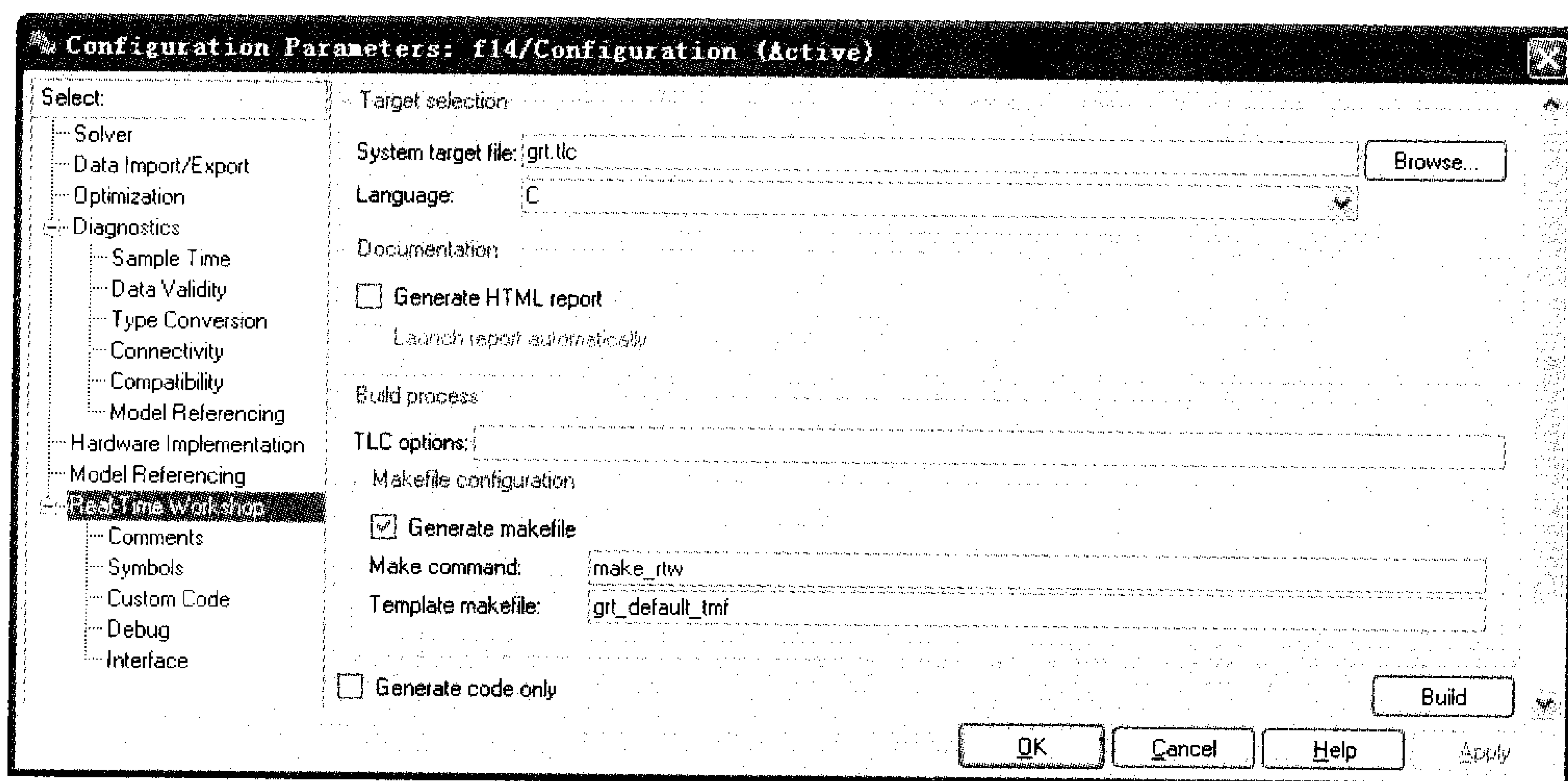


图 6-126　Real-Time Workshop 参数设置

模型参数设置完毕之后，就可以进入建立普通实时程序的环节了。

2）建立程序

模型参数设置完毕后，请读者单击 Real-Time Workshop 中的【Build】按钮，生成 f14 模型的 C 代码。build 命令调用目标文件 grt.tlc，它使用指定的模板文件 rt_default_tmf 来生成 makefile，然后 RTW 再利用生成的 makefile 来建立程序。

当 build 命令执行之后，RTW 的执行过程如下。

（1）编译模型，生成 model.rtw 文件（本例为 f14.rtw）。

（2）调用目标语言编译器，依次编译 TLC 程序，从 grt.tlc 开始，处理 model.rtw（本例为 f14.rtw），并生成代码。

（3）通过模板 makefile 文件（如 grt_vc.tmf）建立名称为 model.mk（本例为 f14.mk）的 makefile 文件。

（4）如果 Simulink 在模板 makefile 文件所指定的宿主机上运行，那么实时程序就生成好了。否则，在生成代码后处理就中断，除非用户在模板 makefile 定义和目标相匹配的宿主机。

在模板 makefile 文件里定义的变量 HOST 用来识别用户的目标系统。HOST 的取值有 3 个选项：PC、UNIX 和 ANY。其中 ANY 的目标系统可以是 DSP 或者微处理器等。

build 命令一旦被执行，RTW 默认情况下就会生成下列文件，这些文件位于系统$\My Documents\MATLAB 目录下。

- f14.c：单机 C 代码。
- f14.h：包含状态变量信息的包含头文件。
- f14_export.h：包含输出信号和参数的包含头文件。
- f14.reg：一个包含头文件，它包含了完成在生成代码你对数据结构进行初始化的模型注册函数信息。
- f14.prm：一个包含模型 f14 中使用的参数的有关信息的包含头文件。
- f14.exe（PC 机上）或 f14（UNIX）：普通的实时可执行代码。

同时在 MATLAB 命令窗口中也会显示相关的信息。部分信息如下：

```
### Starting Real-Time Workshop build procedure for model: f14
（......省略部分信息）
### Generating code into build directory: D:\My Documents\MATLAB\f14_grt_rtw
（......省略部分信息）
### Invoking Target Language Compiler on f14.rtw
        tlc
        -r
        D:\My Documents\MATLAB\f14_grt_rtw\f14.rtw
        D:\Program Files\MATLAB\R2007a\rtw\c\grt\grt.tlc
        -OD:\My Documents\MATLAB\f14_grt_rtw
        -ID:\Program Files\MATLAB\R2007a\rtw\c\grt
        -ID:\My Documents\MATLAB\f14_grt_rtw\tlc
        -ID:\Program Files\MATLAB\R2007a\rtw\c\tlc\mw
        -ID:\Program Files\MATLAB\R2007a\rtw\c\tlc\lib
        -ID:\Program Files\MATLAB\R2007a\rtw\c\tlc\blocks
        -ID:\Program Files\MATLAB\R2007a\rtw\c\tlc\fixpt
        -ID:\Program Files\MATLAB\R2007a\stateflow\c\tlc
（......省略部分信息）
### Loading TLC function libraries.
### Initial pass through model to cache user defined code.
### Caching model source code
### Writing header file f14_types.h
### Writing header file f14.h
```

```
### Writing source file f14.c
### Writing header file f14_private.h
### Writing header file rtmodel.h
### Writing source file f14_data.c
### Writing header file rt_nonfinite.h
### Writing source file rt_nonfinite.c
### TLC code generation complete.
### Creating project marker file: rtw_proj.tmw.
### Processing Template Makefile: D:\Program Files\MATLAB\R2007a\rtw\c\grt \grt_lcc.tmf
### Creating f14.mk from D:\Program Files\MATLAB\R2007a\rtw\c\grt\grt_lcc.tmf
### Building f14: .\f14.bat
（......省略部分信息）
*** Created executable: f14.exe
### Successful completion of Real-Time Workshop build procedure for model: f14
```

3）数据记录

RTW 在产生实时代码过程中，使用 MAT-file 数据记录功能来保存系统状态、输出及仿真时间。为此，请在 Data Import/Export 页的 Save to Workshop 栏选中 Time、States 和 Outputs 选项，使 Real-TimeWorkshop 记录仿真时间、任何离散或者连续系统的状态及其输出。Data Import/Export 页的参数设置如图 6-127 所示。

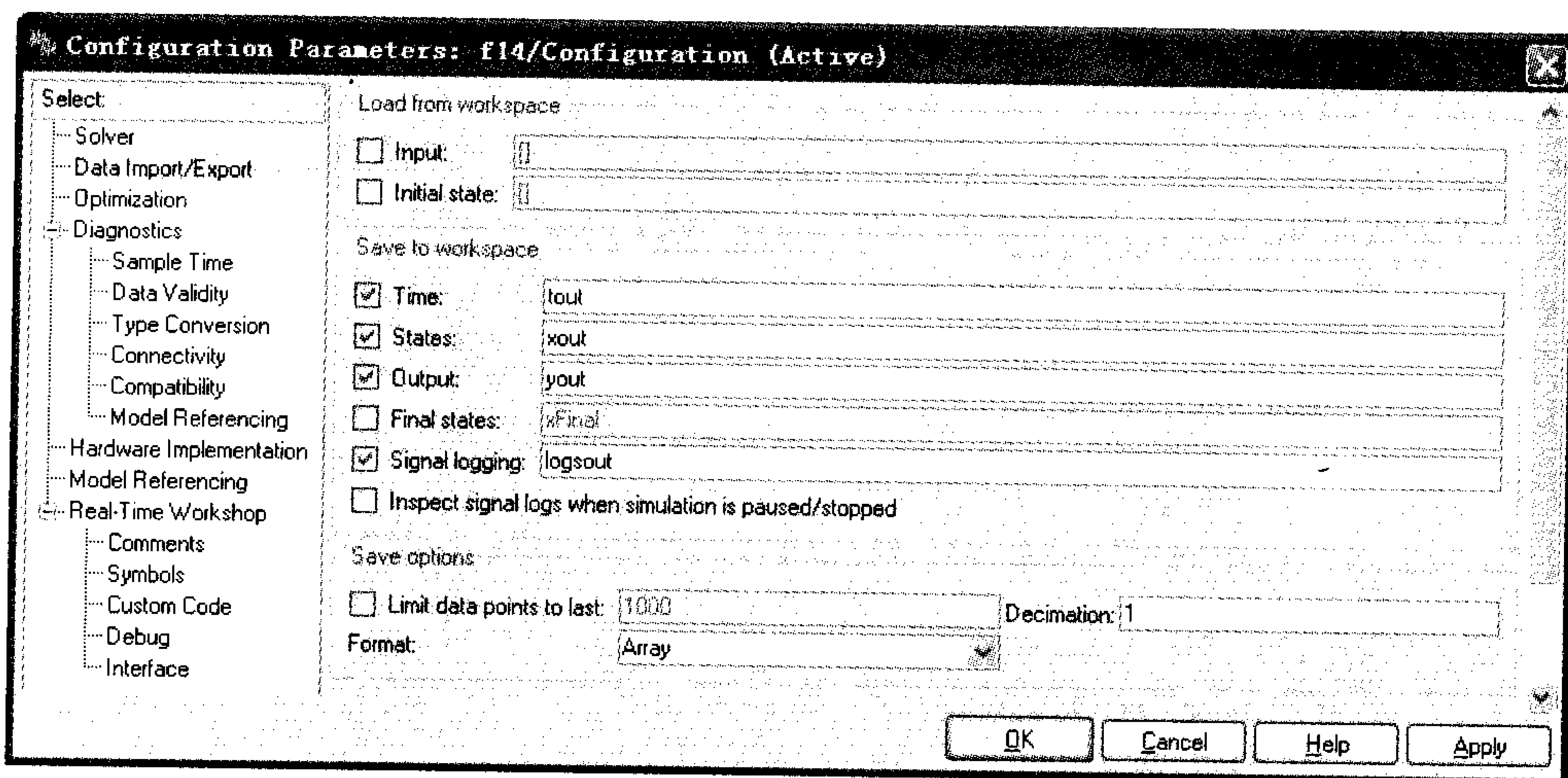


图 6-127　Data Import/Export 的参数设置

此外，如果 Scope 模块设置了保存数据到工作空间属性，那么 RTW 也会将该模块的输入数据保存到 MATLAB MAT 文件（本例为 f14.mat）中。具体操作是，双击 Scope 模块，在其“Parameters”设置对话框 Data history（数据历史）页选中 Save data toworkspace（保存数据到工作空间）选项，然后再在文本框里输入欲存储数据的变量名称，如图 6-128 所示。其中，Stick Input 的变量名为 Stick_Input，另外两个 Scope 的变量名分别为 Angle_of_Attack 和 Pilot_G_force。

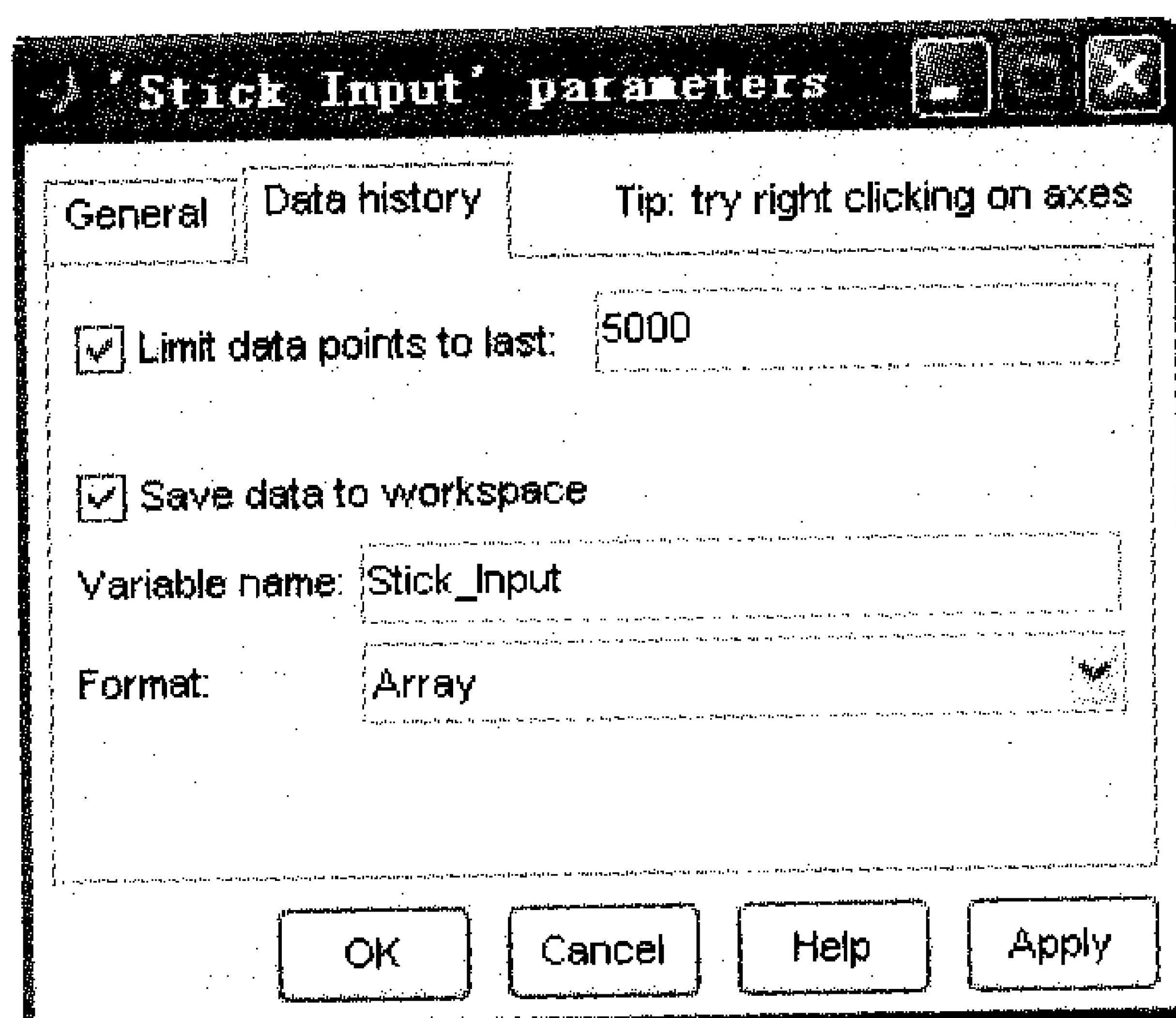


图 6-128 Stick Input 模块的参数设置

如此设置之后，就可以重新用 build 命令来生成可执行代码。然后请读者在命令窗口输入命令：

```
>> !f14     %执行生成可执行的程序
** starting the model **
** created f14.mat **
>>clear     %清除 MATLAB 工作空间已有的变量，以免混淆。
>>load f14
>> who
Your variables are:
rt_Angle_of_Attack        rt_Stick_Input        rt_xout
rt_Pilot_G_force          rt_tout               rt_yout
```

据上文可知，f14.mat 文件里包含了我们刚才设置的 6 个数据变量，但是它们都加上了前缀 rt_，rt_是 real time 的简写。当然，也可以自定义前缀。

4）设置模板 makefile

可以选择以下两种 makefiles 模板。

- UNIX 平台中的 grt_unix.tmf。
- PC 平台中的 grt_vc.tmf、grt_watc.tmf 和 grt_bc.tmf，可以根据不同的编译器来选择相应的模板。

这些文件位于$\MATLAB\R2007a\rtw\c\grt 目录下。和 S-函数的模板一样，用户可以把模板 makefile 复制到当前工作目录，然后进行修改，来改变建立可执行程序的过程。

在 PC 平台的 makefiles 模板中，grt_vc.tmf、grt_watc.tmf 和 grt_bc.tmf 分别是为使用 Microsoft Visual C/C++、Watcom 和 Borland 编译器的用户设计的。它们在使用之前，必须保证相应的编译器已经安装，并设置好相应的环境变量。

5）实时运行接口

在建立实时程序过程中，除了根据模型生成相应的代码之外，还会产生一系列的接口文件。这些文件包括以下几个方面。

- 主程序。
- 驱动模块执行的代码。
- 实现积分算法的代码。

- 实现数据记录的代码。
- 生成 SimStruct 数据结构的代码，SimStruct 用于管理模型的执行。

6）依赖于绝对时间的模块

某些 Simulink 模块使用绝对时间值（它是指仿真系统时间，而不是运行仿真所用的时间）来计算输出。如果用户正在设计一个有可能会一直运行的程序，那么用户不能使用对绝对时间有依赖性的模块。原因很简单，在时间的值达到了 double 精度数字所能表示的最大值时，问题就出现了。在这时，时间就不再增加，于是模块的输出就不再正确了。依赖于绝对时间的模块如表 6-20 所示。

表 6-20　依赖绝对时间的模块

模块名称	子模块名称
连续模块	Derivative Variable，Transport Delay
离散模块	Discrete Time Integrator（在触发子系统）
非线性模块	Rate Limiter
接收器	To Workspace（仅在保存的格式具有时间结构），Scope，To File
源模块	Chirp Signal Generator，Dicrete Pulse Generator，Signal Generator，Clock Ramp，From Workspace Sine Wave，Digital Clock Repeating Sequence，From File Step Code Validation

此外，在仿真参数对话框的 Data Import/Export 页选中 time 选项也依赖于绝对时间。

3．代码验证

在完成了可执行程序的建立之后，需要将 f14 的单机仿真程序和其 Simulink 模型进行比较。在比较之前，我们必须注意以下两点。

- 首先确信 Simulink 和 Real Time Workshop 建立过程采用相同的积分机制（必须是固定步长），例如都是 ode5（Dormand-Prince），同时也要把固定步长设置为相同的数值，如都是 0.05。
- 设置 Scope 模块的 Save data to workspace 选项，以保存 f14 模型的输入和输出数据。例如设置攻击的飞行角 Scope 模块变量为 Angle_of_attack，飞行员所受重力 Scope 模块变量为 Pilot_G_Force，控制输入 Scope 模块变量为 Stick_input。

1）Simulink 模型仿真结果

在 MATLAB 命令窗口中输入命令：

```
>> clear;    % 清除 MATLAB 工作空间的变量
>> f14
```

运行 f14 的 Simulink 模型。对于模型仿真后的变量结果，读者可以在 MATLAB 命令窗口中查看。

2）调用普通实时程序运行结果

在上节“实时代码生成方法与步骤”的“数据记录”中，已经产生了相应的单机程序运行结果。这里，我们通过 MATLAB 命令窗口，直接调用其结果数据。数据调用的命令如下：

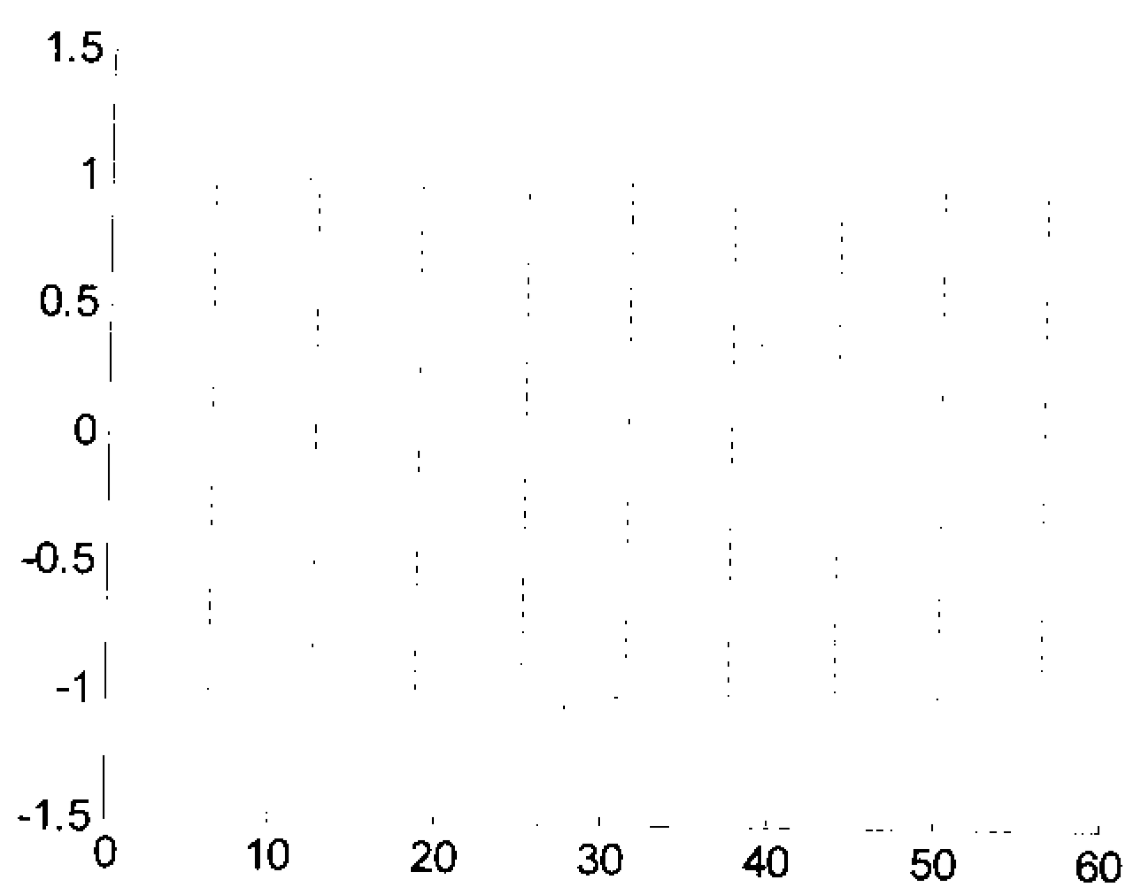
```
>>load f14
```

3）Simulink 模型和普通实时程序结果比较

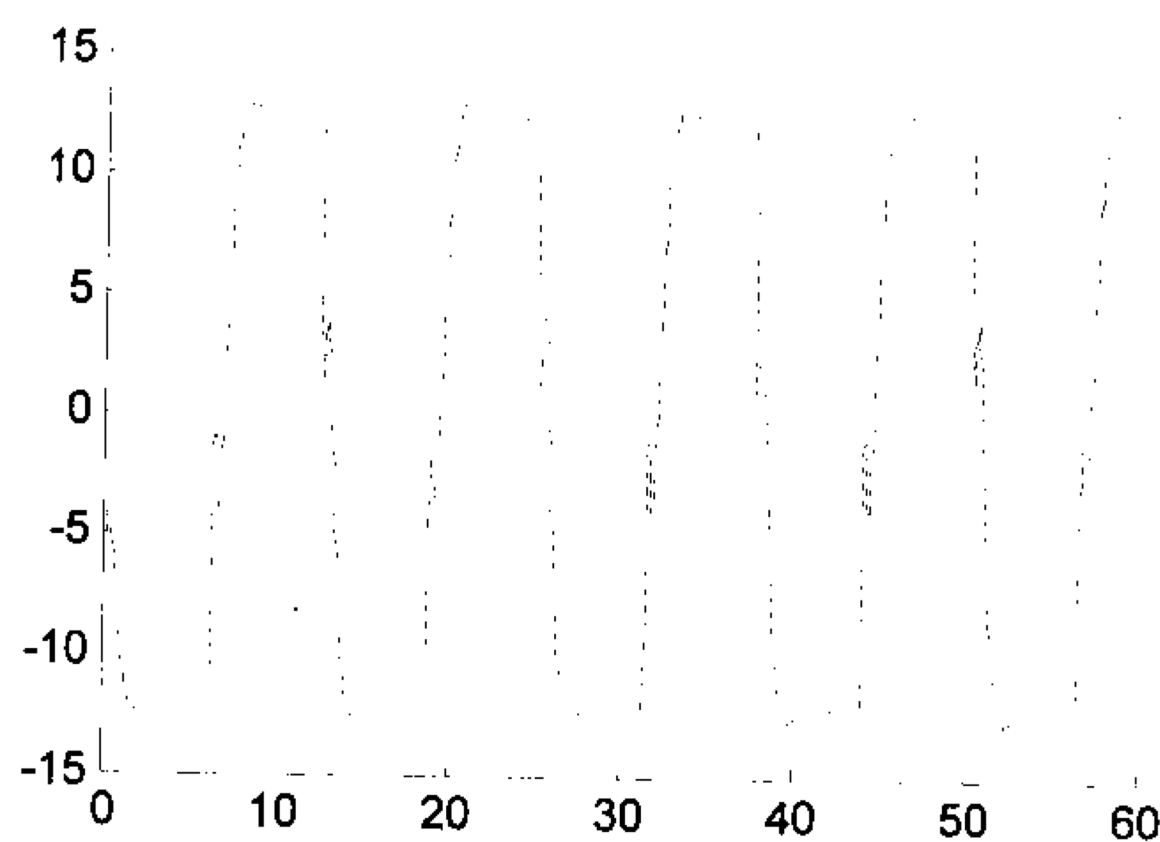
首先我们通过绘制曲线图，使读者对 Simulink 模型仿真结果和单击程序运行结果的比

较有个感性的认识。图形的绘制名如下，绘制结果如图 6-129 所示。据图可知，参数 Stick_Input 和 rt_Stick_Input、Pilot_G_force 和 rt_Pilot_G_force，以及 Angle_of_Attack 和 rt_Angle_of_Attack 的波形无明显差别，初步说明了 Simulink 模型和普通实时程序运行结果的正确性。

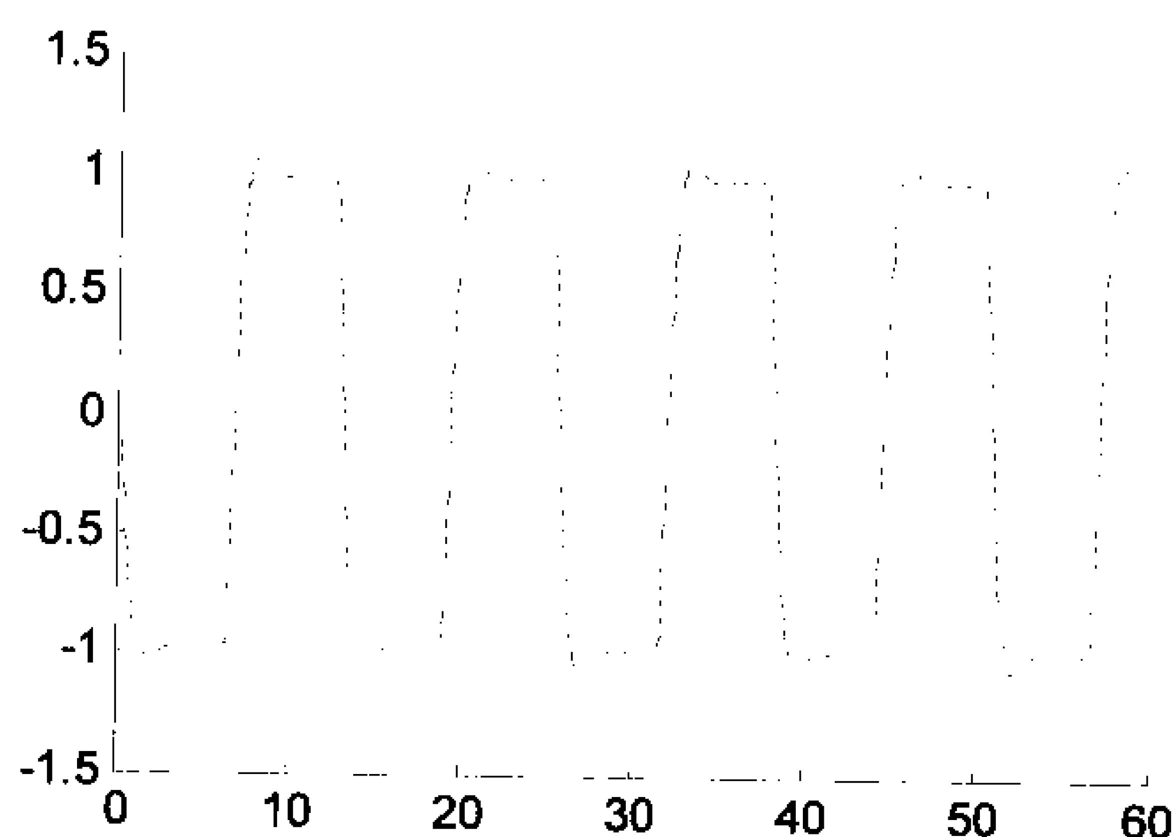
```
>> plot(rt_tout, [Stick_Input(:,2) rt_Stick_Input(:,2)]);
>> figure;
>> plot(rt_tout, [Pilot_G_force(:,2) rt_Pilot_G_force(:,2)]);
>> figure;
>> plot(rt_tout, [Angle_of_Attack(:,2) rt_Angle_of_Attack(:,2)]);
```



（a）Stick_Input 和 rt_ Stick_Input 波形



（b）Pilot_G_force 和 rt_Pilot_G_force 波形



（c）Angle_of_Attack 和 rt_Angle_of_Attack 波形

图 6-129 Simulink 模型和普通实时程序运行结果比较

为了量化比较这 3 种参数之间的差别，我们可以采用如下方法。

（1）比较 Stick_Input 和 rt_Stick_Input，得到：

```
>>max(abs(rt_Stick_Input - Stick_Input))
ans =
     0     0
```

（2）比较 Pilot_G_force 和 rt_Pilot_G_force，得到：

```
>>max(abs(rt_Pilot_G_force - Pilot_G_foce))
ans =
     0     0
```

（3）比较 Angle_of_attack 和 rt_Angle_of_attack，得到：

```
>> max(abs(rt_Angle_of_attack - Angle_of_attack))
ans =
```

0 0

可以看出，在默认的精度下，Simulink 模型和普通实时程序运行结果一致。

6.7.3 Simulink 模型实时代码生成方法与实例

上节讲述了 Simulink 模型的普通实时程序的生成方法，但是在有些时候，我们只需要产生 Simulink 模型的实时代码，不需要模型进行编译和连接。本节主要通过实例，讲述只生成 Simulink 模型实时代码的方法，熟悉 Real-Time Workshop 提供的优化性能。

例 6.17 建立正弦信号增益调节模型，并生成该模型的实时代码。

1．Simulink 模型建立

建立的模型如图 6-130 所示。该模型用于实现对正弦信号增益的调节，并通过 Scope 显示仿真结果，将模型的文件名保存为 exa07_17.mdl。

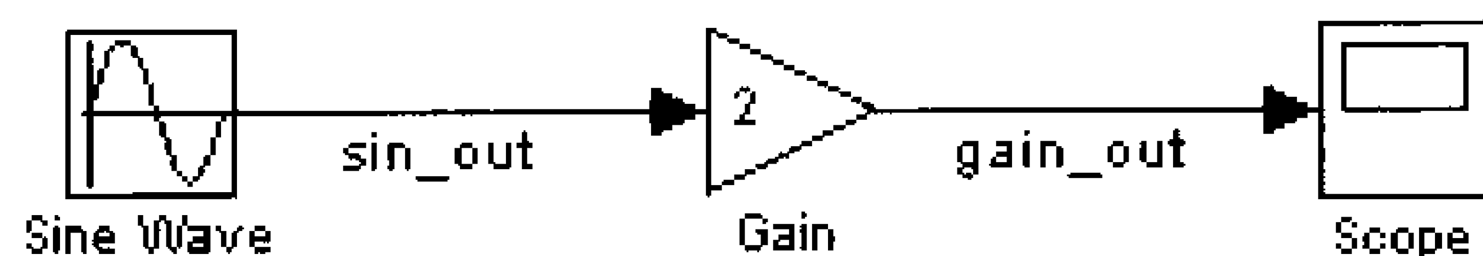


图 6-130 实时代码生成的 Simulink 模型

2．模型参数设置

设置模型基本参数的步骤如下。

（1）在 Simulink 窗口中，执行【Simulation】→【Configuration Parameters】命令。

（2）在模型参数设置对话框 Solver 项中，设置 Type 为 Fixed-step，Solver 为 ode5(Dormand-Prine)。

（3）打开 Data Import/Export 页，除 Signal logging 选项之外，其他所有选项都为非选中状态。

（4）打开 Real-Time Workshop 页，选中 Generate code only 复选框。当 Generate code only 复选框选中后，这时【Build】按钮就会变成【Generate code】按钮，同时，System target file 编辑框中为 grt.tlc。

（5）确定模型参数设置，并保存模型。

3．不用缓冲优化生成代码

当开启模块的优化功能时，RTW 都会使用缓存来保存模块的输出数据。当然，我们也可以通过以下操作，取消模块输出数据缓存的功能。

（1）打开 Configuration Parameters 对话框中的 Optimization 项，不选中 Signal storage reuse 选项和 Implement logic signal as boolean data(vs.double)选项，如图 6-131 所示。

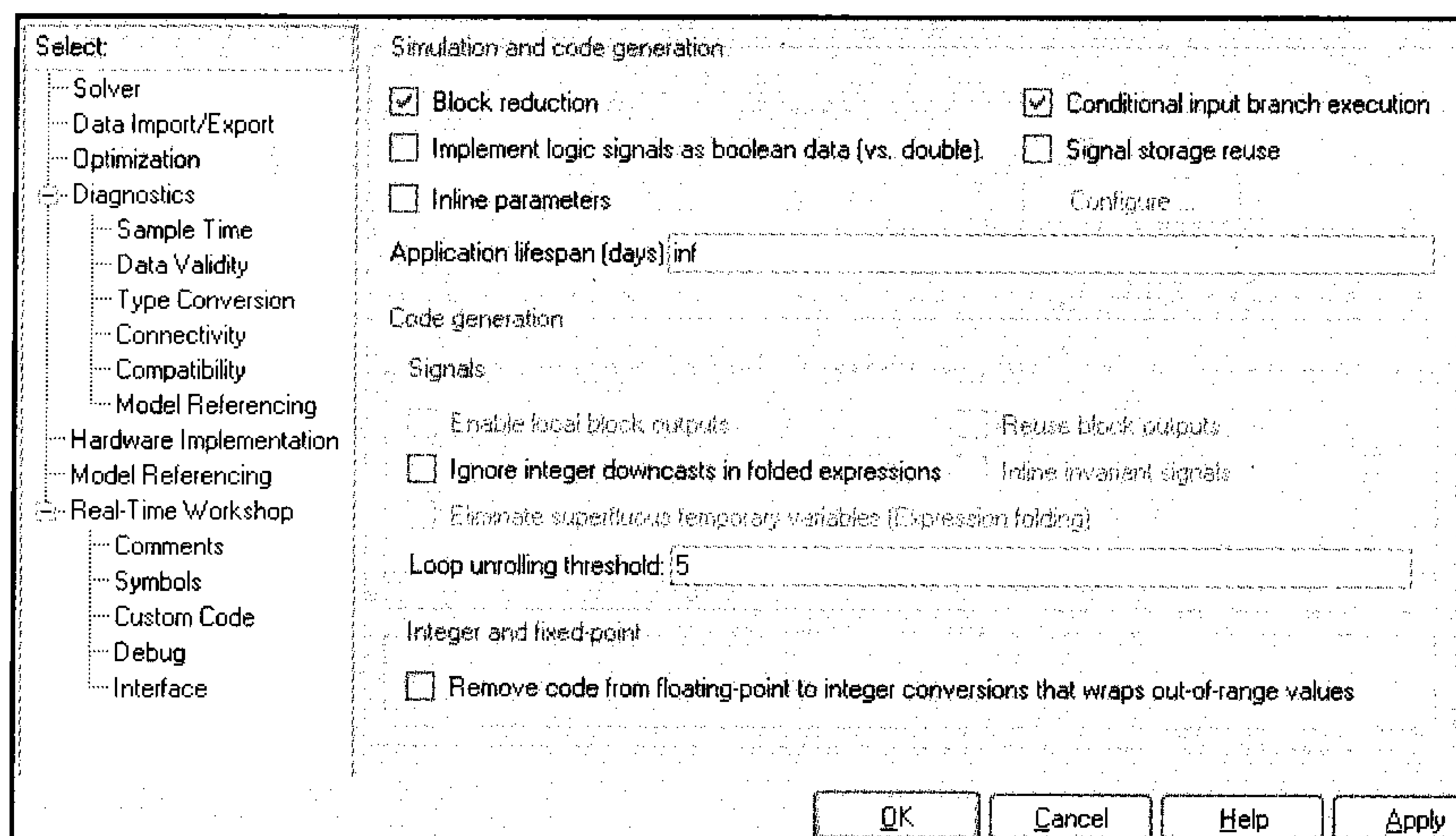


图 6-131　模型代码生成优化参数设置

（2）打开 Real-Time Workshop 页，单击【Generate code】按钮。

在 MATLAB 窗口会产生如下代码生成信息：

```
## Starting Real-Time Workshop build procedure for model: exa07_17
### Generating code into build directory: E:\Matlab\exa07_17_grt_rtw
（……省略部分信息）
### Invoking Target Language Compiler on exa07_17.rtw
        tlc
        -r
        E:\Matlab\exa07_17_grt_rtw\exa07_17.rtw
        d:\Program Files\MATLAB\R2007a\rtw\c\grt\grt.tlc
        -OE:\Matlab\exa07_17_grt_rtw
        -Id:\Program Files\MATLAB\R2007a\rtw\c\grt
        -IE:\Matlab\exa07_17_grt_rtw\tlc
（……省略部分信息）
### Loading TLC function libraries.
### Initial pass through model to cache user defined code
### Caching model source code
### Writing header file exa07_17_types.h
### Writing header file exa07_17.h
### Writing source file exa07_17.c
### Writing header file exa07_17_private.h
### Writing header file rtmodel.h
### Writing source file exa07_17_data.c
### Writing header file rt_nonfinite.h
### Writing source file rt_nonfinite.c
### TLC code generation complete.
### Creating project marker file: rtw_proj.tmw.
### Processing Template Makefile: d:\Program Files\MATLAB\R2007a\rtw\c\grt\ grt_lcc.tmf
### Creating exa07_17.mk from d:\Program Files\MATLAB\R2007a\rtw\c\grt\grt_lcc.tmf
```

```
### Successful completion of Real-Time Workshop build procedure for model: exa07_17
```

（3）产生所有代码包含在系统自动创建的 exa07_17_grt_rtw 目录中，在 MATLAB 命令窗口输入命令：

```
>> edit exa07_17_grt_rtw\exa07_17.c
```

打开模型计算结果文件 exa07_17.c。

（4）查看 exa07_17.c 文件中的 exa07_17_output 函数，我们发现由于没有使用数据缓冲优化功能，exa07_17_output 函数为 exa07_17_B.sin_out 和 exa07_17_B.gain_out 分配了同一个缓冲单元。

（5）在 GRT 兼容调用接口中，exa07_17_output 被封装函数 MdlOutputs 调用，代码如下：

```
void MdlOutputs(int_T tid)
{
  exa07_17_output(tid);
}
```

4．使用缓冲优化生成代码

（1）打开 Optimization 项，选中 Signal storage reuse 复选框，而不选中 Implement logic signal as boolean data(vs.double)复选框。

（2）此时图 6-131 中 Code generation 选项组 Enable local block outputs，Reuse block outputs 和 Eliminate superfluous temporary variables (Expression folding)处于标记状态，但不能操作其状态。

（3）打开 Real-Time Workshop 项，单击 Generate codc 按钮。

（4）用 MATLAB 编辑器打开 exa07_17.c 文件，并查看 exa07_17_output 函数，发现它只为 Gain 模块的计算结果直接合并到模型输出中，不需要分配临时存贮单元。也就是说，使用缓冲优化技术之后，提高了代码的生成效率。

5．代码生成结果的 HTML 报告

在模型参数设置对话框中，当 Real-Time Workshop General 项中 Generate HTML report 和 Launch report automatically 的复选框被选中时，在代码生成完毕之后，就会弹出一个带有导航栏的源代码报告对话框，如图 6-132 所示。

在 HTML 报告中，用户可以单击相应的链接打开需要查看的源文件，HTML 报告一个非常好的作用在 Summary 栏中可以知道产生代码的时间，当单击 click to open 链接时，会弹出 Simulink 模型参数设置对话框（不过是只读状态），可以知道模型参数的设置情况。

用户在任何时候都可以查看 HTML 报告，该文件名为 exa07_17_codegen_rpt.html，位于 exa07_17_grt_rtw\html\目录下。

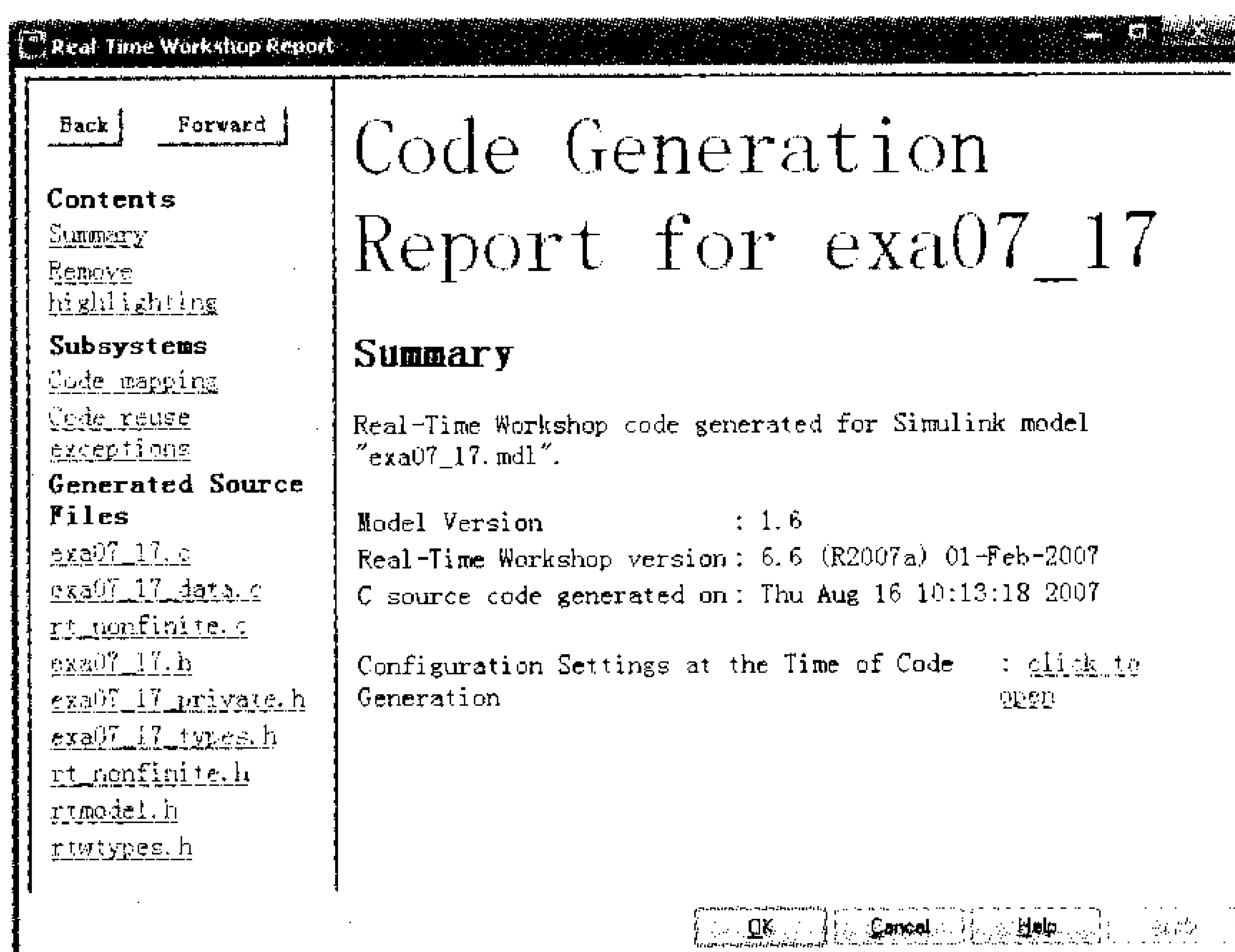


图 6-132　模型代码生成结果的 HTML 报告

第7章 MATLAB的工程应用

MATLAB是一种用于概念设计、算法开发、建模仿真、实时实现的科学计算软件，它将高性能的数值计算和可视化方法集成在一起，并提供了很多专业工具箱和大量的内置函数，已经被广泛地应用于科学计算、控制系统、信息处理等领域的分析、仿真和设计工作，这和它强大的工具箱是分不开的，第1章已经提到，MATLAB本身提供了许多面向工程应用的专用工具箱，用户能够方便快捷地使用，免除了自己编写复杂而庞大的算法程序。通过灵活调用这些工具箱的功能函数，就可以完成比较复杂的专业应用，来解决具体工程应用中的算法设计问题，这为工程仿真和开发提供了一条非常便捷和高效的渠道。

掌握MATLAB的工程应用非一日之功，因为它涵盖的内容非常宽泛，包含的函数非常多，而实际问题通常也是千变万化的。因此，本章的真正目的并不在于让读者掌握MATLAB在这些工程领域中的全部功能，而是通过实例做一些示范性的介绍，介绍一些函数的调用方法和一些问题的解决途径，读者在学习时要注重体会MATLAB的强大功能和解决问题的基本流程，通过举一反三达到无师自通的学习效果，而没有必要也不太可能记住每一个函数的功能和用法。在实际中遇到具体问题时，可先通过帮助系统搜寻是否具有相应的函数，也可通过相应的工具箱进行浏览，找到合适的函数之后，结合具体的帮助文件并结合里面的具体实例来查询和学习它们的调用方法。

本章主要内容：

- MATLAB与信号处理
- MATLAB与图像处理
- MATLAB与控制工程

7.1 MATLAB与信号处理

从常用信号变换、IIR和FIR数字滤波器设计到平稳和非平稳信号分析，MATLAB都具备这些方面强大的处理功能，并且能用图形化的用户界面加以实现。其中，常用信号变换包括Z变换、Chirp Z变换、FFT变换、DCT变换和Hilbert变换等。离散系统结构包括IIR、FIR和Lattice结构。IIR滤波器设计包括模拟和数字低通、高通、带通与带阻滤波器设计，以及基于冲激响应不变法和双线性Z变换法的IIR滤波器设计等。FIR滤波器设计包括基于窗函数、频率抽样法和切比雪夫逼近法的FIR滤波器设计。平稳信号分析包括经典功率谱估计、基于参数模型的功率谱估计和基于非参数模型的功率谱估计。非平稳信号分析包括STFT变换、Gabor展开、Wigner-Ville分布与Choi-Williams分布。非高斯信号分析包括基于非参数法的双谱估计、基于参数模型的双谱估计，以及双谱估计的应用。信号处理的图形工具实现包括滤波器设计与分析的FDATool工具和滤波器设计与信号分析的SPTool工具。

7.1.1 MATLAB 实现信号变换

利用 MATLAB 可以实现的变换有 Z 变换、Chirp Z 变换、离散傅里叶变换（DFT）、离散余弦变换（DCT）以及 Hilbert 变换等。以下给一个 Hilbert 变换和 Chirp Z 变换的例子。

MATLAB 工具箱提供了计算 Hilbert 变换的函数 hilbert.m，其格式如下：

```
y=Hilbert(x)
```

但需注意的是，该函数计算出的结果 y 的实部为序列本身，其虚部才是序列的 Hilbert 变换。

例 7.1　$x(n)$ 为正弦序列，其频率为 $f=\dfrac{1}{16}$，长度为 25，求其 Hilbert 变换 $\hat{x}(n)$。

例 7.1 的 MATLAB 实现如例程 7.1 所示，其运行结果如图 7-1 所示。

例程 7.1　序列的 Hilbert 变换

```
clear all;
N=25;
f=1/16;
x=sin(2*pi*f*[0:N-1]);
y=hilbert(x);
% 求 x 的希尔伯特变换;
subplot(211)
stem(x,'.');
hold on;
plot(zeros(size(x)));
title('原始序列')
subplot(212)
stem(imag(y),'.');
% imag(y) is the Hilbert transform of x(n);
title('序列 Hilbert 变换结果')
```

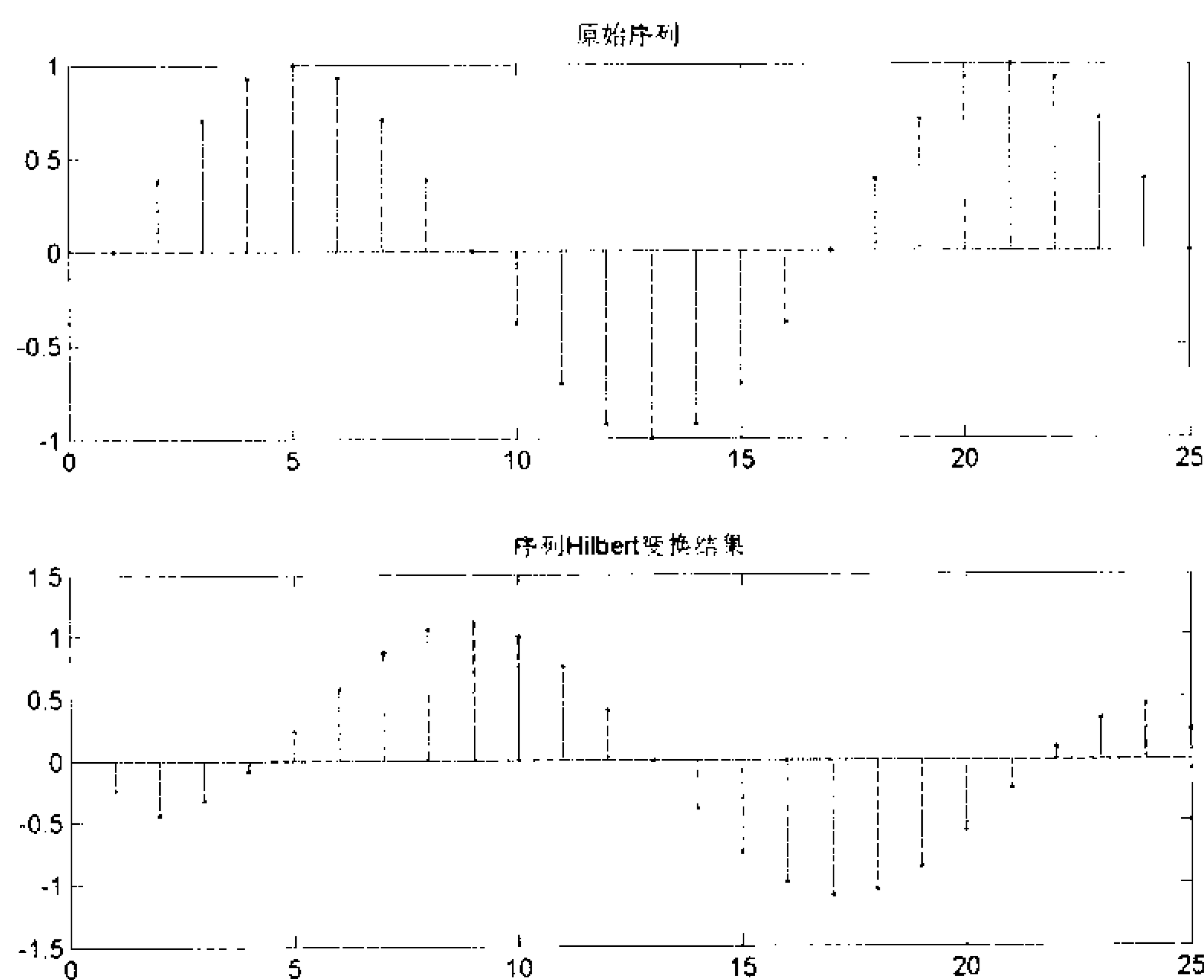


图 7-1　序列的 Hilbert 变换

在 MATLAB 中实现线性调频 Z（Chirp Z）变换非常简单，只需调用工具箱中的 czt 函

数即可。该函数的调用格式如下：

```
y=czt[x,m,w,a]
```

它用来计算序列 *x* 沿着由 *w* 和 *a* 定义的螺旋线上的 Z 变换。*m* 指定变换长度，*w* 指定 *z* 平面螺旋线上的点之间的比率，*a* 指定起始点。当 *m*、*w*、*a* 未指定时，其相当于 FFT。

例 7.2　设序列 $x(n)$ 由 3 个正弦函数组成，频率分别为 8Hz，8.22Hz 和 9Hz，抽样频率为 40Hz，时域取 128 点，比较序列 CZT 和 FFT 的计算结果。

例 7.2 的 MATLAB 实现如例程 7.2 所示，其运行结果如图 7-2 所示。

例程 7.2　序列的 CZT 及和 FFT 的比较

```
clear all;
% 构造三个不同频率的正弦信号的叠加作为试验信号
N=128;
f1=8;
f2=8.22;
f3=9;
fs=40;
stepf=fs/N;
n=0:N-1;
t=2*pi*n/fs;
n1=0:stepf:fs/2-stepf;
x=sin(f1*t)+sin(f2*t)+sin(f3*t);
M=N;
W=exp(-j*2*pi/M);

% A=1 时的 CZT 变换
A=1;
Y1=czt(x,M,W,A);
subplot(311)
plot(n1,abs(Y1(1:N/2)));
ylabel('A=1 时的 CZT')
grid on;

% FFT
Y2=abs(fft(x));
subplot(312)
plot(n1,abs(Y2(1:N/2)));
ylabel('FFT')
grid on;

% 详细构造 A 后的 CZT
M=60;
f0=7.2;
DELf=0.05;
A=exp(j*2*pi*f0/fs);
W=exp(-j*2*pi*DELf/fs);
Y3=czt(x,M,W,A);
n2=f0:DELf:f0+(M-1)*DELf;
subplot(313);
```

```
plot(n2,abs(Y3));
ylabel('详细构造 A 后的 CZT')
grid on;
```

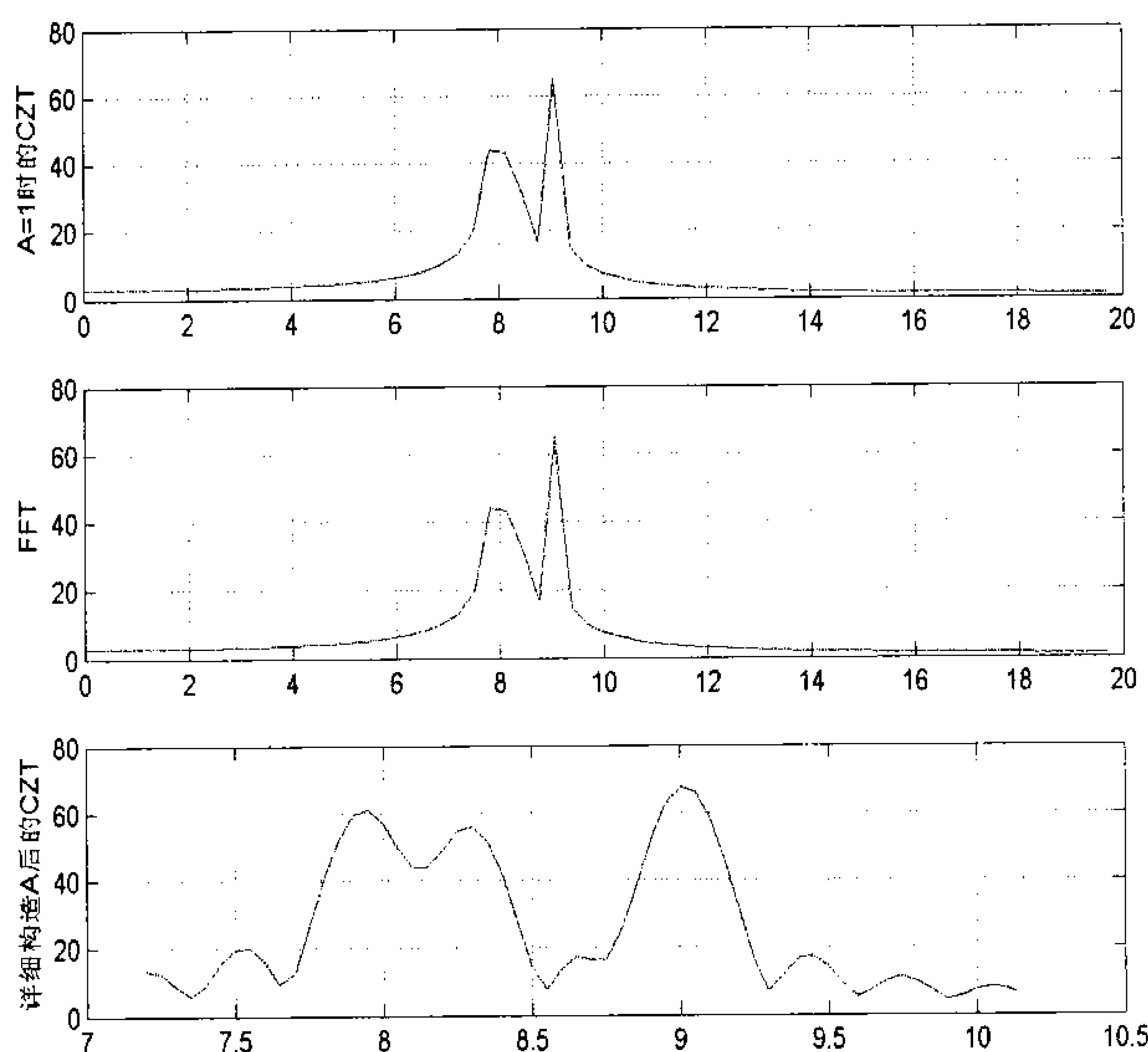


图 7-2　序列的 CZT 和 FFT 比较

从图 7-2 可以看出，A=1 的 CZT 变换的结果与 FFT 变换一样，不能区分信号 8Hz 和 8.22Hz 的频率成分。当详细构造 A 之后，在 $7 \sim (7+M\times0.05)$ 频段范围内求序列 $x(n)$ 的 CZT，信号中的 3 种频率成分的谱线都可以分辨出来。

7.1.2　MATLAB 实现数字滤波

由于信号往往夹杂噪声及无用信号成分，所以必须将这些干扰成分滤除，滤波器可以对信号进行筛选，只让特定频段或者满足特点要求的信号通过。滤波运算是信号处理中的基本运算。因此，滤波器的设计问题也是数字信号处理中的基本问题。

利用 MATLAB 实现 IIR 数字滤波的方法有很多，比如于巴特沃斯法、切比雪夫法、椭圆法、Yule-Walk 法、Prony 法、线性预测法、Steiglitz-McBride 法以及反向频率法直等。MATLAB 工具箱中提供了设计数字滤波器的函数，使 IIR 数字滤波器的设计变得简单，这些函数有 butter、cheby1、cheby2、ellip、Yule-Walk、Prony、lpc、stmcb，以及 infreqz。以下给出一个 Yule-Walk 法的实现实例。

Yule-Walk 法设计数字滤波器实际上是一种递归的数字滤波器设计，该函数只能进行数字滤波器的设计，不能进行模拟滤波器的设计。它在频域上采用最小均方算法进行设计。在 MATLAB 中，实现该方法的函数为 Yule-Walk，它的调用格式如下：

```
[b,a] = YuleWalk(N,f,m)
```

该函数返回 Yule-Walk 滤波器系统传递函数分子与分母系数向量 b 和 a，它们的阶次为 N+1。其中向量 f 和 m 表示理想滤波器的幅频特性，f 为归一化的频率向量，该向量中每一元素都在 0 到 1 之间取值，而且元素必须是递增排序，并要求第一个元素为 0，最后一个

元素为 1（采样频率的一半）。*m* 是对应 *f* 频率处的幅度，它也是一个向量，并且向量的长度和 *f* 相同。

当确定理想滤波器的幅度频率响应后，为了避免通带到阻带的陡峭过渡，应该对过渡带进行多次的试验，以便得到最佳的滤波器。

例 7.3　利用 Yule-Walk 方法设计一个 10 阶的低通数字滤波器，对应的理想滤波器的截止频率为 300Hz，抽样频率为 1000 Hz。

例 7.3 的 MATLAB 程序如例程 7.3 所示，该数字滤波器的幅度与相位特性曲线及冲激响应如图 7-3 所示。

例程 7.3　Yule-Walk 数字低通滤波器设计

```
clear all;
f = [0 0.6 0.6 1];
m = [1 1 0 0];
[b,a] = yulewalk(10,f,m);
[h,w] = freqz(b,a,256);
plot(f,m,'-',w/pi,abs(h),'*')
text(0.7,1.2,'-:理想频率响应')
text(0.7,1.1,'*:实际频率响应')
```

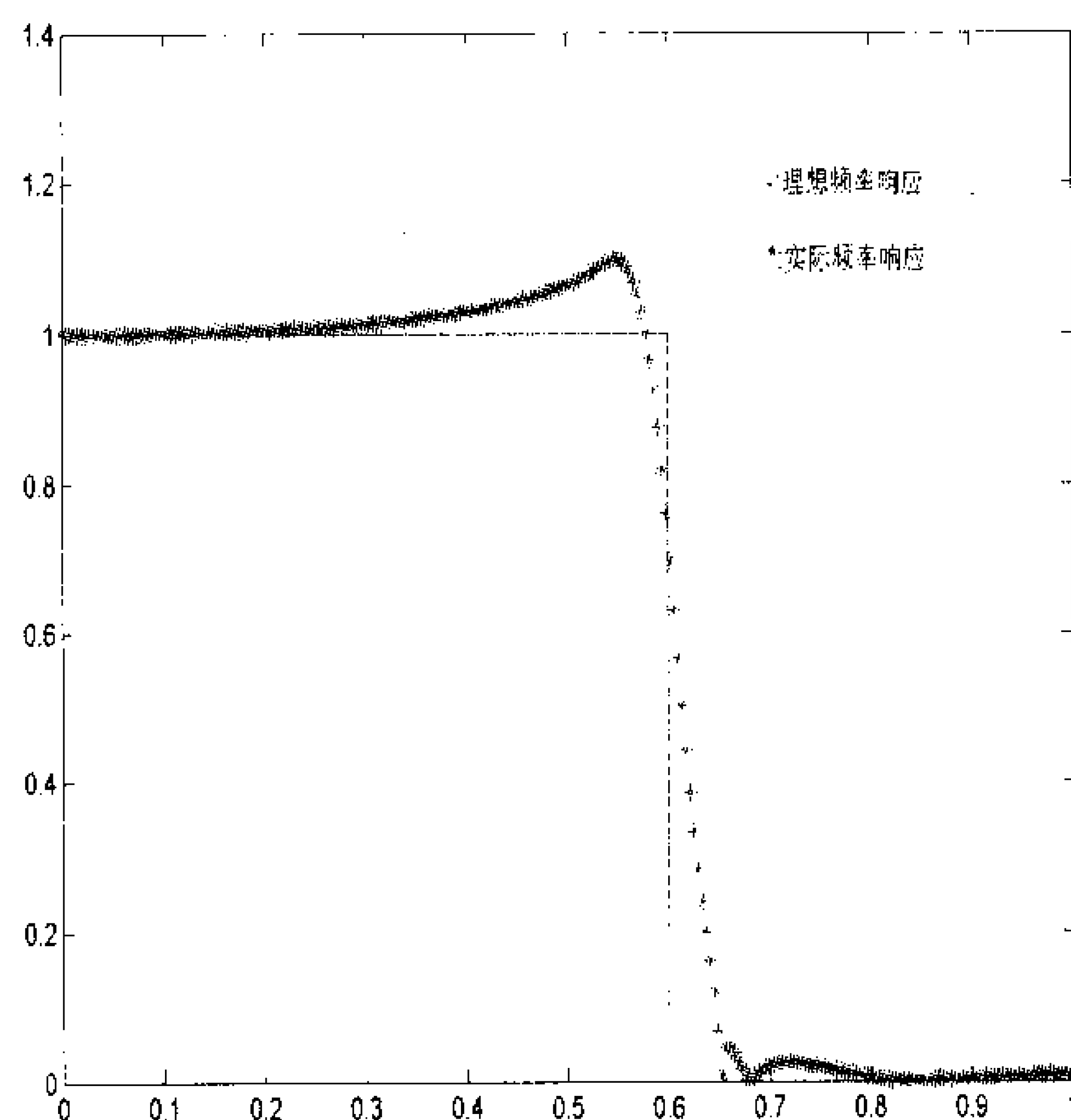


图 7-3　Yule-Walk 数字低通滤波器的频率响应曲线

7.1.3　MATLAB 实现功率谱估计

功率谱估计方法可以分成经典谱估计法与现代谱估计法。经典谱估计法又可分为直接法与间接法，直接法是利用快速傅里叶变换 FFT 算法对有限个样本数据进行傅里叶变换得到功率谱的方法，又称为周期图法；间接法是先得到样本数据的自相关函数估计，然后进行傅里叶变换得到功率谱的方法。现代谱估计的提出主要是针对经典谱估计的分辨率低和方差性能不好等问题提出的。从现代谱估计的方法上，大致可分为参数模型谱估计和非参数模型谱估计。参数模型谱估计主要包括 AR 模型、MA 模型、ARMA 模型，以及最小方差谱估计等；非参数模型谱估计主要有基于矩阵特征值分解的功率谱估计，即特征向量谱

估计与 MUSIC 方法谱估计。

这些功率谱的估计均可以通过 MATLAB 实现。以下给出利用函数 periodogram 实现直接法的功率谱估计：

例 7.4 序列 $x(n)=\exp\left(\mathrm{j}\omega_0 n-\mathrm{j}\pi\right)+\exp\left(\mathrm{j}\omega_1 n-\mathrm{j}0.7\pi\right)+e(n)$ 为复正弦加白噪声的平稳信号，其中 $\omega_0=100\pi$，$\omega_1=50\pi$，$e(n)$ 为零均值的白噪声，信噪比 $S/N=10\mathrm{dB}$。要求：（1）产生仿真数据；（2）利用直接法估计序列的功率谱。

该例的 MATLAB 实现如例程 7.4 所示，序列 $x(n)$ 及其功率谱估计曲线图如图 7-4 所示。

例程 7.4 序列 $x(n)$ 的功率谱估计

```
clear all;
Fs=1000;%采样频率
%产生含有噪声的序列
var=sqrt(1/exp(1.0));
n=0:1/Fs:1;
N=length(n);
e=var*randn(1,N);
w0=100*pi;
w1=50*pi;
xn=exp(j*w0*n-j*pi)+exp(j*w1*n-j*0.7*pi)+e;

%绘制信号波形
subplot(311)
plot(n,abs(xn))
xlabel('n')
title('x(n)=exp(j*w0*n-j*pi)+exp(j*w1*n-j*0.7*pi)+e(n)')

%计算序列的 DFT
nfft=1024;
Xk=fft(xn,nfft);
%计算序列的 PSD
Pxx1=abs(Xk).^2/N;
%绘制功率谱图形
index=0:round(nfft/2-1);
k=index*N/nfft;
plot_Pxx1=10*log10(Pxx1(index+1));
subplot(312)
plot(k,plot_Pxx1)
ylabel('公式直接计算的功率谱')

%periodogram 函数计算的功率谱
window=boxcar(length(xn));
[Pxx2,f]=periodogram(xn,window,nfft,Fs);
plot_Pxx2=10*log10(Pxx2(index+1));
subplot(313)
plot(k,plot_Pxx2)
xlabel('periodogram 函数计算的功率谱')
```

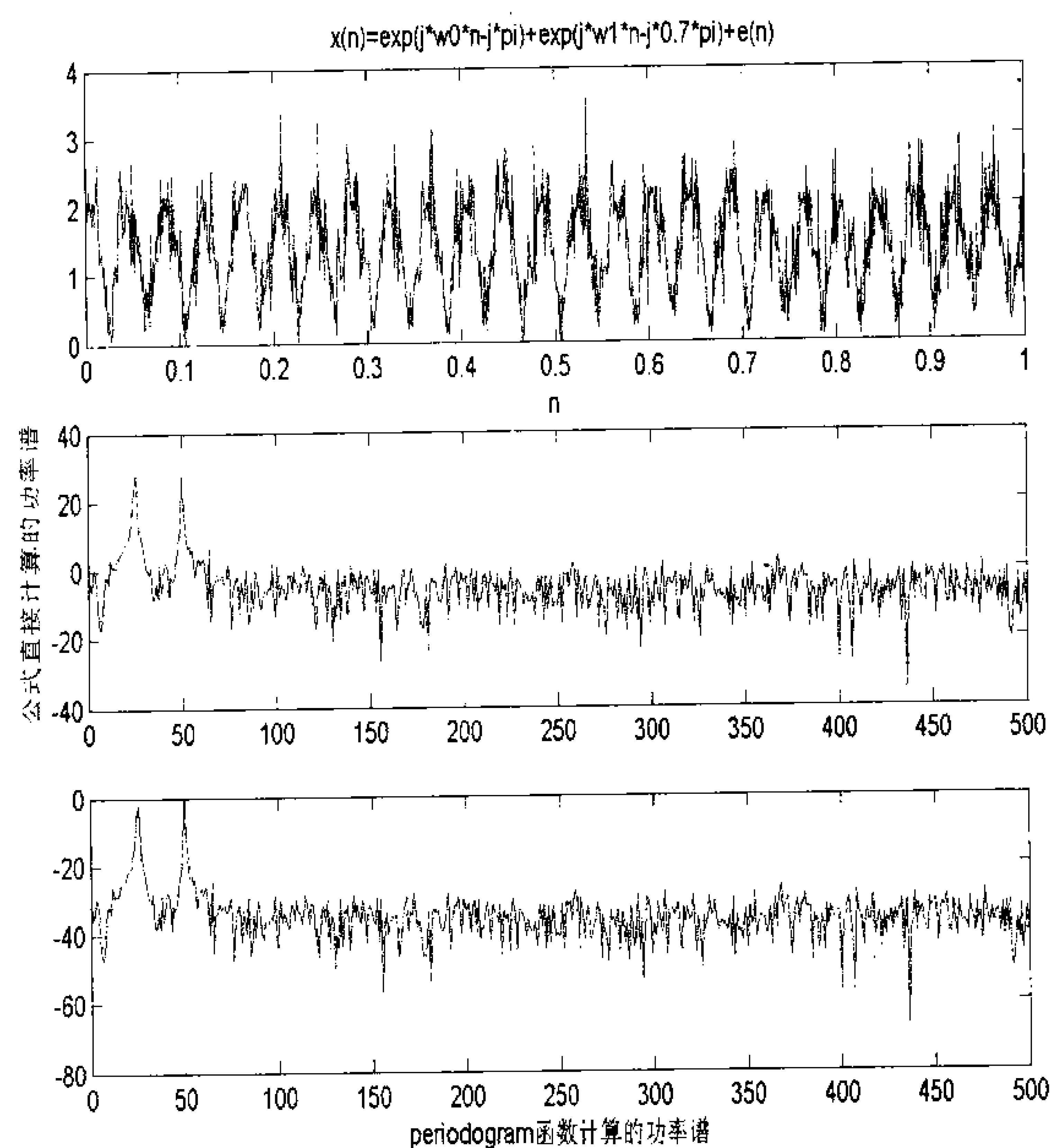


图 7-4 序列 $x(n)$ 的直接法功率谱估计曲线

从图 7-4 中可以看出利用公式直接计算的序列 $x(n)$ 的功率谱与利用函数 periodogram 计算得到的功率谱完全一致。

7.1.4 小波变换在语音信号处理中的应用

小波变换（Wavelet Transform）的概念是 1984 年法国地球物理学家 J.Morlet 在分析处理地球物理勘探资料时提出来的。小波变换的数学基础是 19 世纪的傅里叶变换，其后理论物理学家 A.Grossman 采用平移和伸缩不变性建立了小波变换的理论体系。1985 年，法国数学家 Y.Meyer 第一个构造出具有一定衰减性的光滑小波。1988 年，比利时数学家 I. Daubechies 证明了紧支撑正交标准小波基的存在性并成功构造了它，使得离散小波分析成为可能。1989 年 S. Mallat 提出了多分辨率分析概念，统一了在此之前各种构造小波的方法，特别是提出了二进离散小波变换的快速算法，使得小波变换走向实用性。

小波分析方法是一种窗口大小（窗口面积）固定但其形状可改变，时间窗和频率窗都可改变的时频局域化分析方法，即在低频部分具有较高的频率分辨率和较低的时间分辨率，在高频部分具有较高的时间分辨率和较低的频率分辨率，所以被称为数学显微镜。正是这种特性，使小波变换具有对信号的自适应性。

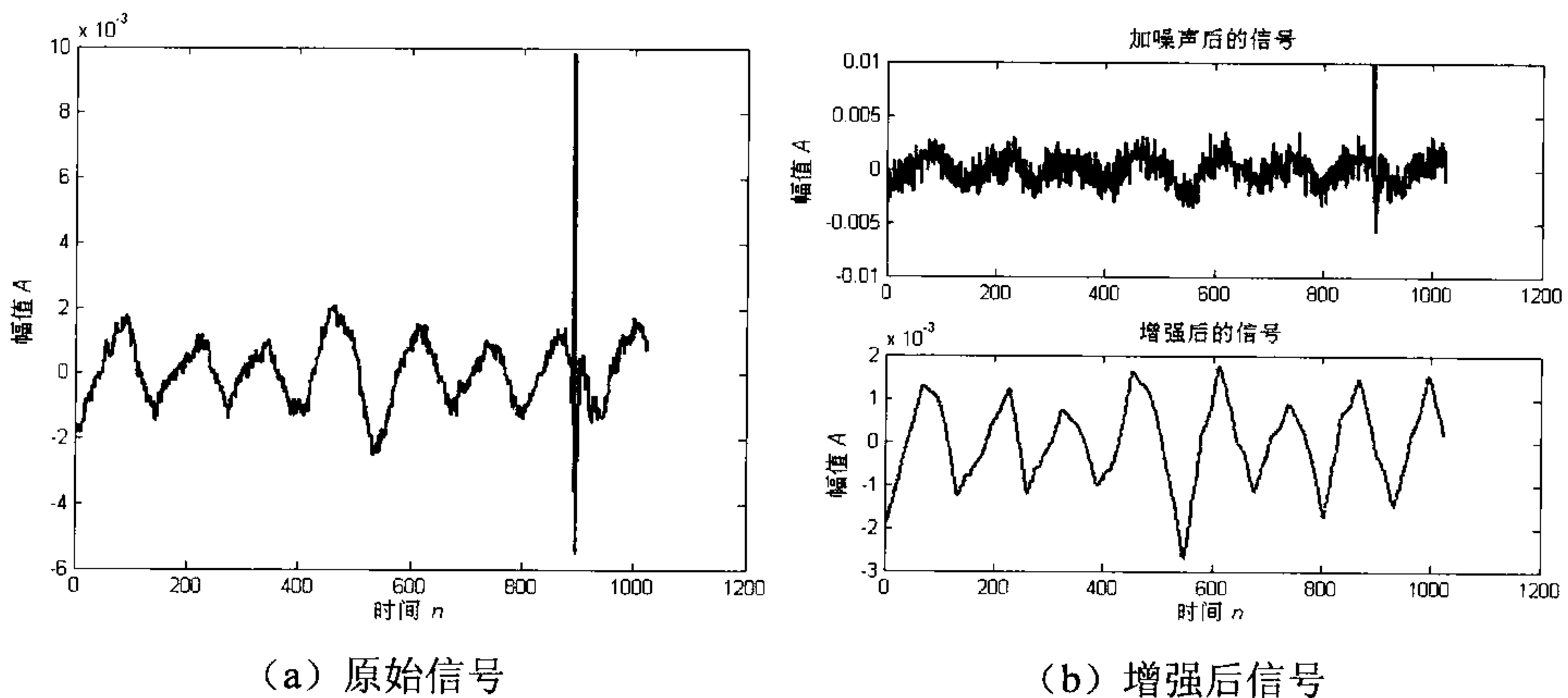
因此小波分析理论受到众多学科的共同关注，近十多年来，小波分析的理论和方法在信号处理、语音分析、模式识别、数据压缩、图像处理、数字水印、量子物理等专业和领域得到了广泛的应用。以下给出一个小波分析应用于语音信号处理的实例。

例程 7.5 利用小波变换增强语音信号

```
%装载语音信号
N=1024;
```

```
s=wavread('fem.wav',N);
figure(1);
plot(1:N,s,'LineWidth',2);
xlabel('时间 n');ylabel('幅值 A');
s=s+0.001*randn(1,N)';
%用小波 db3 对 s 进行 5 层分解
level=5;
[c,l]=wavedec(s,level,'db3');
%选用全局阈值进行信号增强处理
thr=5;
[sd,csd,lsd,perf0,perfl2]=wdencmp('gbl',c,l,'db3',level,thr,'h',1);
figure(2);
subplot(2,1,1); plot(s,'LineWidth',2);
title('加噪声后的信号');
%xlabel('时间 n');
ylabel('幅值 A');
subplot(2,1,2);
plot(sd,'LineWidth',2); title('压缩后的信号');
xlabel('时间 n');ylabel('幅值 A');
```

原始的语音信号波形如图 7-5（a）所示，可以看见语音信号中含有一定的噪声。为使本例更有说服力，本例程首先对原始语音信号追加噪声，然后对其利用小波变换进行分解，最后重构后得到的信号如图 7-5（b）所示，从图中明显可以看出，增强后的语音信号很光滑，基本不含任何噪声分量。



（a）原始信号　　　　（b）增强后信号

图 7-5　原始语音信号波形和增强后的语音信号波形

7.1.5　MATLAB 实现 SAR 信号处理

运动目标成像问题是合成孔径雷达（SAR）信号处理的一个难题，常规 SAR 的距离向高分辨率是通过线性调频信号的脉冲压缩技术获得的，方位向的高分辨率是通过接受数据与一个理论上的静止目标的冲击响应作匹配相关获得的。因此如果成像场景中有以前未知方式运动的目标，常规 SAR 的成像方法就不能正常工作，从而造成运动目标图像的模糊和方位偏移。

SAR 信号的频率是随时间连续变化的，因此它是一个典型的非平稳信号，可以利用时频分析方法提取瞬时频率曲线，而且时频分析方法并不需要知道关于目标的先验知识，它能同时完成检测和参数估计。

SAR 静止目标的后向散射信号可以表示为：

$$S(x)=a(x)\exp\{-jkx^2/R_0\}，\quad x\in V_aT$$

式中 T 为合成孔时间，$k=2\pi/\lambda$，λ 为工作波长，$a(x)$ 是天线方位向权函数，V_a 是雷达平台速度，R_0 是目标与雷达平台之间的距离。

如果目标是运动的，并设 (V_r,a_r) 是运动目标的径向速度和加速度，(V_c,a_c) 是运动目标的方位向的速度和加速度，则散射信号可以表示为：

$$S_m=a[(1-\varepsilon_c^{'})x]\exp(-2jk\varepsilon_r^{'}x)\exp\{-jkx^2[(1-\varepsilon_c^{'})^2-\varepsilon_r^{''}]/R_0\}$$

式中 $\varepsilon_r^{'}=V_r/V_a,\varepsilon_c^{'}=V_c/V_a,\varepsilon_r^{''}=a_rR_0/V_a^2$。

由此可见，运动目标和静止目标的主要差别在于它们的中心频率和调频率不同。SAR 运动目标回波信号是一种复杂的时变信号，联合时频分析方法是解决这一问题的有效工具。常用的 Wigner-Ville 分布能满足要求，但是它也有致命的缺点，由于它是双线性变换，因此用于多点目标和面目标时会出现交叉项干扰，严重降低了它的性能，为此本小节将利用 Hough 变换来抑制交叉项。

例程 7.6　利用 Wigner-Hough 变换分析两目标的 SAR 信号

```
clear;
%仿真合成孔径雷达信号
colormap(gray(256))
cj=sqrt(-1);
pi2=2*pi;

c=3e8;                  % 传播速度
fc=200e6;               % 频率
lambda=c/fc;            % 波长
k=pi2/lambda;           % 波数
Xc=1.e3;                % 距离目标中心的距离

L=400;                  % 合成孔径大小为 2*L
Y0=100;                 % 目标区域位于[Yc-Y0,Yc+Y0]
Yc=0;                   % 距离目标中心的跨距离

theta_c=atan(Yc/Xc);  %到目标中心的斜视角
Rc=sqrt(Xc^2+Yc^2);    %到目标中心的斜视距离
kus=2*k*sin(theta_c); %Doppler 频率偏移

Xcc=Xc/(cos(theta_c)^2);
```

```
DY=(Xcc*lambda)/(4*L);              % 交叉距离分辨率
L_min=max(Y0,L);                    % 补零的孔径

% 压缩信号的u域参数和阵列
du=(Xcc*lambda)/(4*(Y0+L));         % 孔径域的采样间隔
duc=(Xcc*lambda)/(4*Y0);            %压缩信号孔径域的采样间隔
mc=2*ceil(L_min/duc);
uc=duc*(-mc/2:mc/2-1);
dkuc=pi2/(mc*duc);                  % ku 域的采样间隔
kuc=dkuc*(-mc/2:mc/2-1);            % kuc 阵列
%
dku=dkuc;                           % ku 域的采样间隔

%合成孔径信号的u域参数和阵列
m=2*ceil(pi/(du*dku));              %样本数
du=pi2/(m*dku);
u=du*(-m/2:m/2-1);                  % 合成孔径阵列
ku=dku*(-m/2:m/2-1);                % ku 阵列
%
ntarget=2;                  % 目标数
%目标的坐标和反射率
yn(1)=.7*Y0;            fn(1)=0.5;
yn(2)=yn(1)-4*DY;       fn(2)=1;

%测量的回波信号
s=zeros(1,mc);
for i=1:ntarget;
  dis=sqrt(Xc^2+(Yc+yn(i)-uc).^2);
  s=s+fn(i)*exp(-cj*2*k*dis).*(abs(uc) <= L);
end;
figure(1);
plot(uc,real(s));
xlabel('合成孔径 u');
ylabel('幅值 A');
title('孔径信号的实部');
axis([uc(1) uc(mc) 1.1*min(real(s)) 1.1*max(real(s))]);

sig=s(1:50);
%计算 Wigner-Ville 分布
[tfr,t,f]=tfrwv(sig',1:length(sig),256);
%显示 Wigner-Ville 分布等高线
figure(2);
mesh(t,f,tfr);
xlabel('时间 t');
ylabel('频率 f');
```

```
zlabel('幅值 A');

%Hough 变换
[WH,rho,theta]=htl(tfr);
figure(3);
mesh(rho,theta,WH');
xlabel('极半径 rho');
ylabel('角度 theta');
zlabel('幅值 A');
```

数值仿真生成的合成孔径回波信号的实部波形如图 7-6 所示，从时域图上不能看出目标的存在；计算其 Wigner-Ville 分布如图 7-7 所示，由于没有噪声的干扰，因此在图上也能看出两个目标的存在，但存在有交叉项。

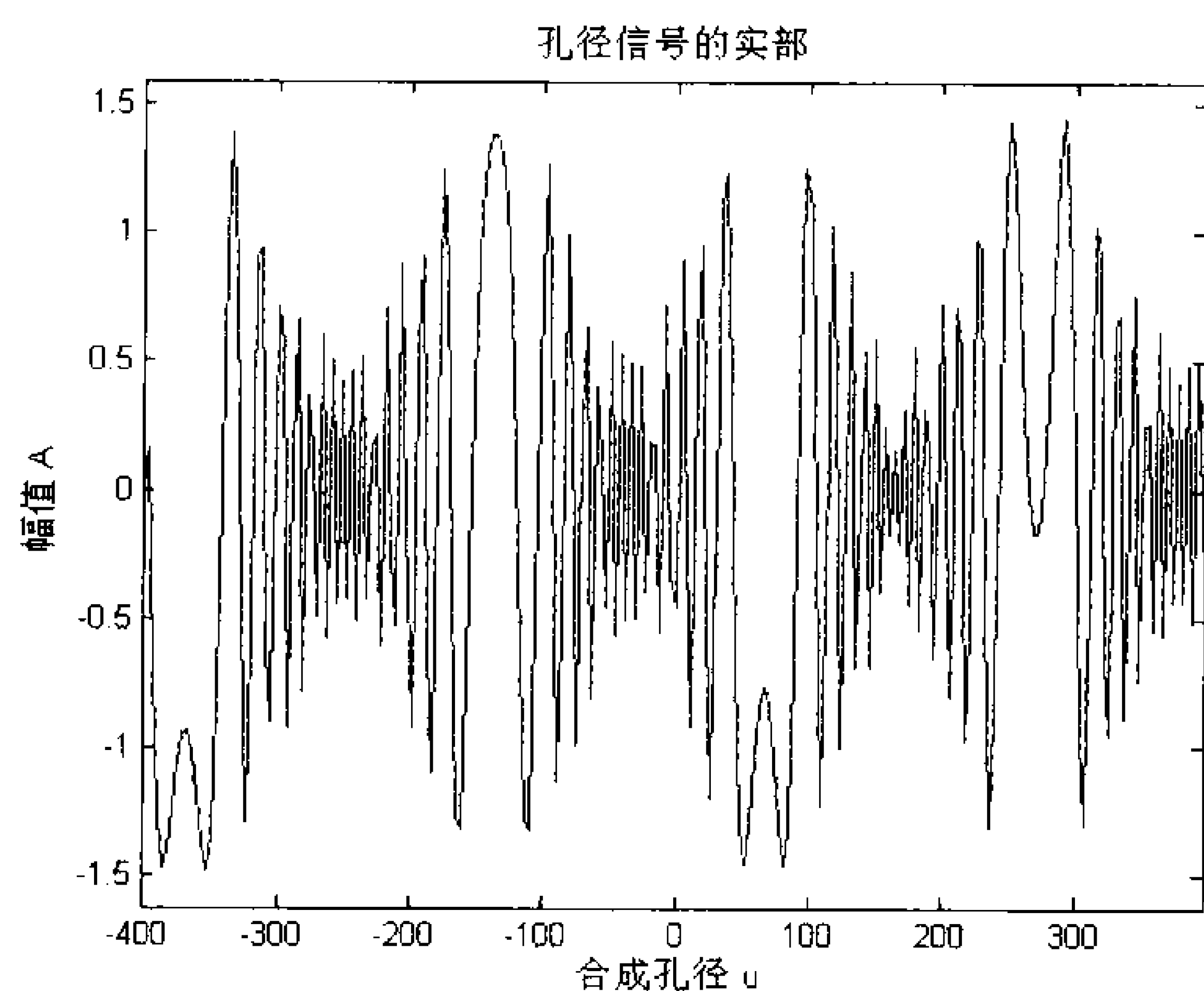


图 7-6 合成孔径回波信号的实部波形

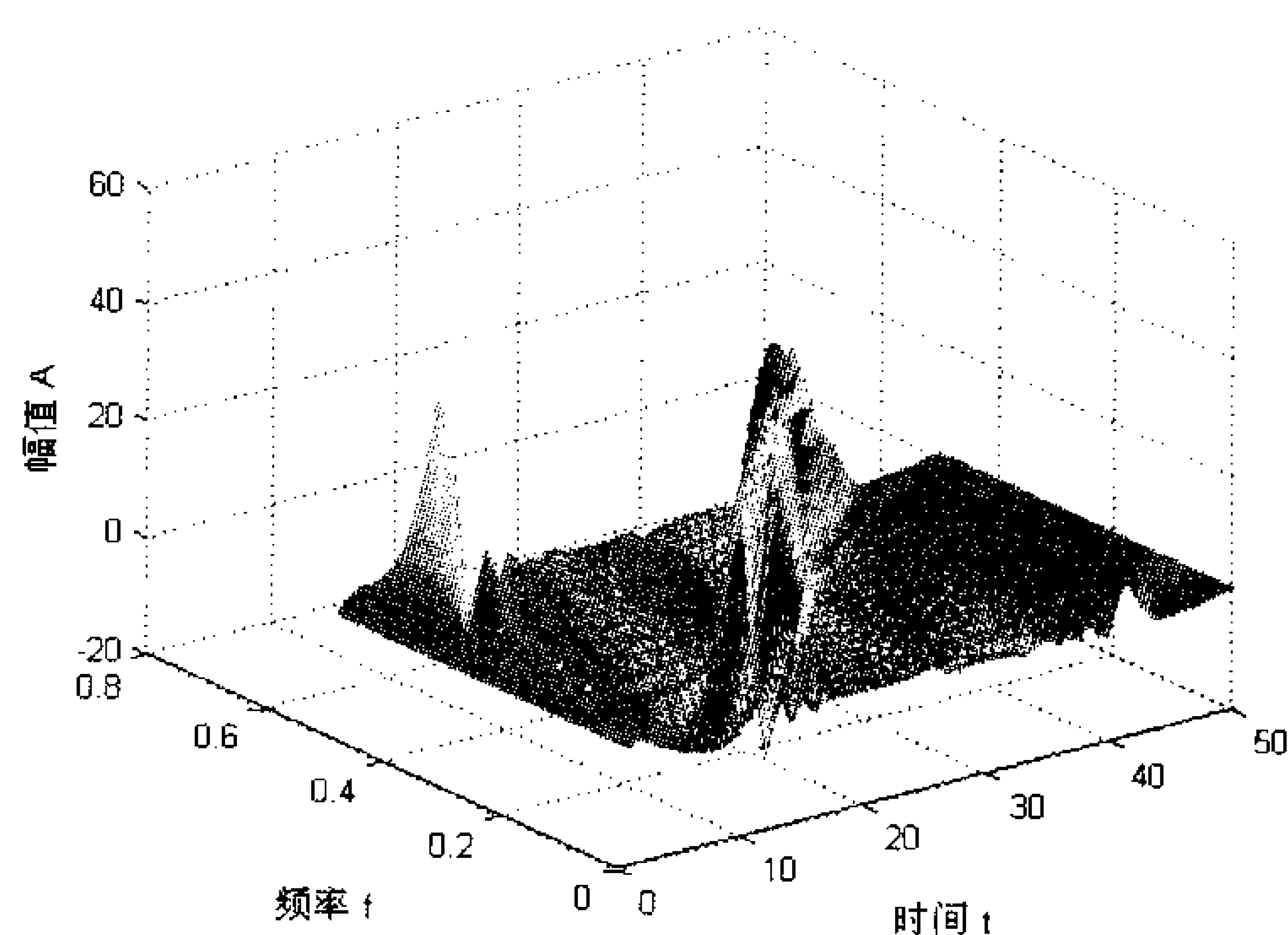


图 7-7 合成孔径回波信号的 Wigner-Ville 分布

进一步对 Wigner-Ville 分布进行 Hough 变换，结果如图 7-8 所示，从图上明显能看出两处存在峰值，它对应了两个面目标。

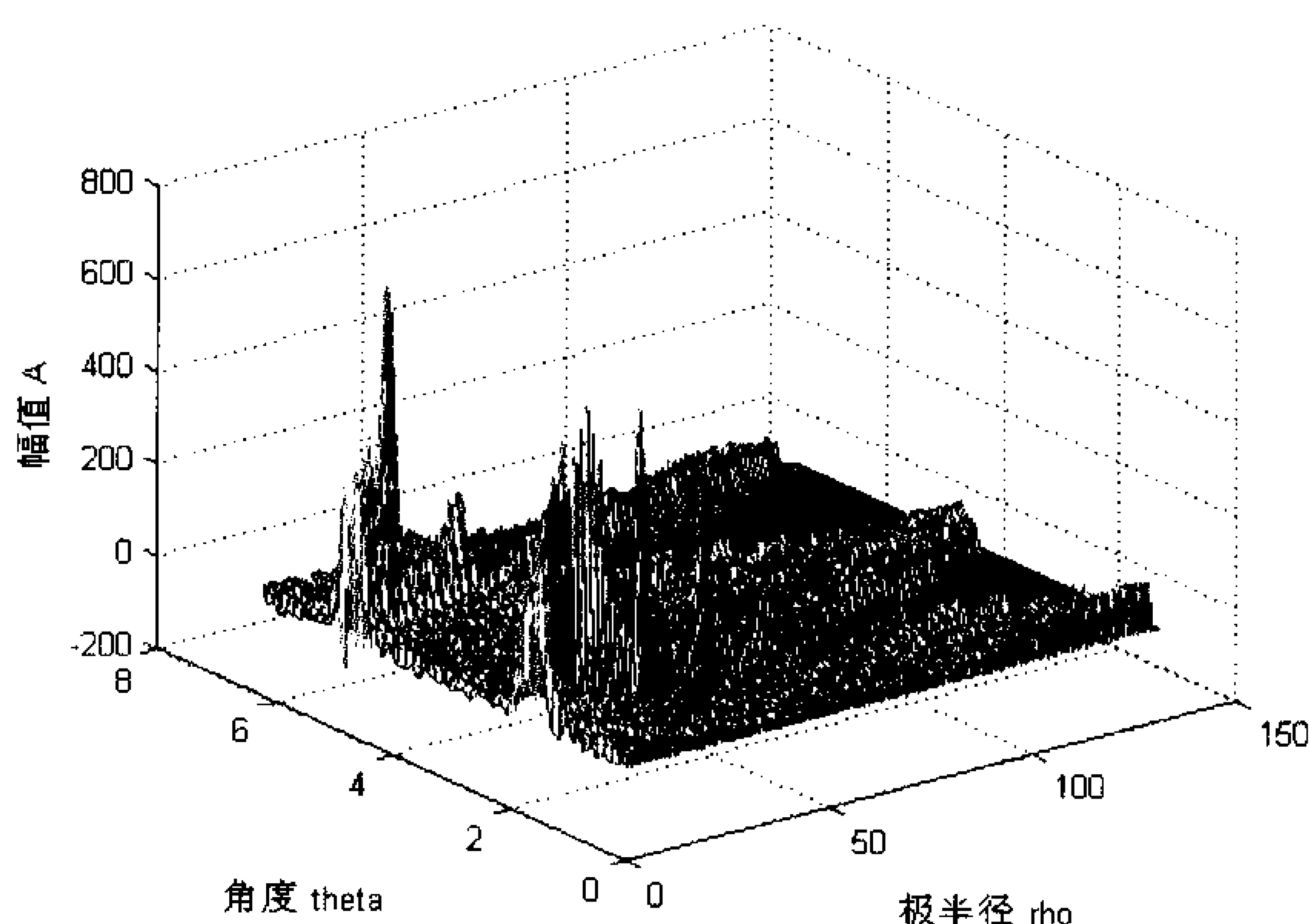


图 7-8 合成孔径回波信号的 W-H 变换

7.2 MATLAB 与图像处理

图像处理工具箱是 MATLAB 环境下开发出来的许多工具箱之一，对应于 R2007a 的工具箱版本为 V5.4，它是以数字图像处理理论为基础，用 MATLAB 语言构造得到的一系列用于图像数据显示与处理的 M 文件。图像处理工具箱可支持的图像处理的范围很广，几乎包含了我们常见的所有图像处理函数，从第一层次的图像变换（包括空域变化）、图像运算、形态学运算，第二层次的图像增强和滤波，到第三层次的图像理解（图像分析和感兴趣区域操作），涵盖了绝大部分图像处理的内容。特别的，它也包含了遥感图像中常用到的内容：图像配准。

此外，对于图像的基本操作，比如读入输出保存等都得到了完美的支持。当然，我们也可以自己编写函数来扩展其功能。MATLAB 中的信号处理工具箱、神经网络工具箱、模糊逻辑工具箱和小波工具箱也用于协助执行图像处理任务。

7.2.1 图像变换

图像变换在数字图像处理与分析中有着很重要的作用，是一种常用的、有效的分析手段。图像变换的目的是使图像处理问题简化，更有利于图像特征提取，有助于从概念上加强对图像信息的理解。通过正交变换与酉变换改变图像的表示域及表示数据给后续的处理工作带来极大的方便；常用的傅氏变换使得图像的处理可以在频域中进行，使运算简便；图像经过变换后往往能反映出图像的灰度结构特征，便于分析；此外，许多变换可以使得能量集中在少数数据上，从而实现数据压缩，便于图像传输和存储。

图像变换的算法很多，包括线性变换、离散傅氏变换、离散余弦变换、Hough 变换和沃尔什-哈达玛变换等。Hough 变换属于特征提取技术，在 MATLAB 中，实现这个功能的函数就是 hough。检测的结果我们可以用 houghline 函数检测 ***H*** 中的叠加点，从而检测出 BW 的线段出来。下面给出一个实例。

例 7.7 用 hough 函数检测图像中的直线，Hough 变换的结果如图 7-9 所示。

```
RGB = imread('gantrycrane.png');
I   = rgb2gray(RGB); % 转化成强度图像
BW = edge(I,'canny'); % 提取边界
[H,T,R] = hough(BW,'RhoResolution',0.5,'ThetaResolution',0.5);
% 显示原始图
subplot(2,1,1);
imshow(RGB);
title('gantrycrane.png');
%% 显示 Hough 矩阵
subplot(2,1,2);
imshow(imadjust(mat2gray(H)),'XData',T,'YData',R,...
        'InitialMagnification','fit');
title('Hough transform of gantrycrane.png');
xlabel('\theta'), ylabel('\rho');
axis on, axis normal, hold on;
```

```
colormap(hot);
%%
P  = houghpeaks(H,5,'threshold',ceil(0.3*max(H(:))));
lines = houghlines(BW,T,R,P,'FillGap',5,'MinLength',7);
figure, imshow(BW), hold on;
max_len = 0;
for k = 1:length(lines)
    xy = [lines(k).point1; lines(k).point2];
    plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');

    % 显示线段的开头和结尾
    plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
    plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');

    % 检测最长险段的终点
    len = norm(lines(k).point1 - lines(k).point2);
    if ( len > max_len)
        max_len = len;
        xy_long = xy;
    end
end

%% 保存图像
imwrite(I,'cha3_8_21.bmp','bmp');
imwrite(BW,'cha3_8_22.bmp','bmp');
H=H/(max(max(H)));
imwrite(H,'cha3_8_23.bmp','bmp');
```



原图



强度图像

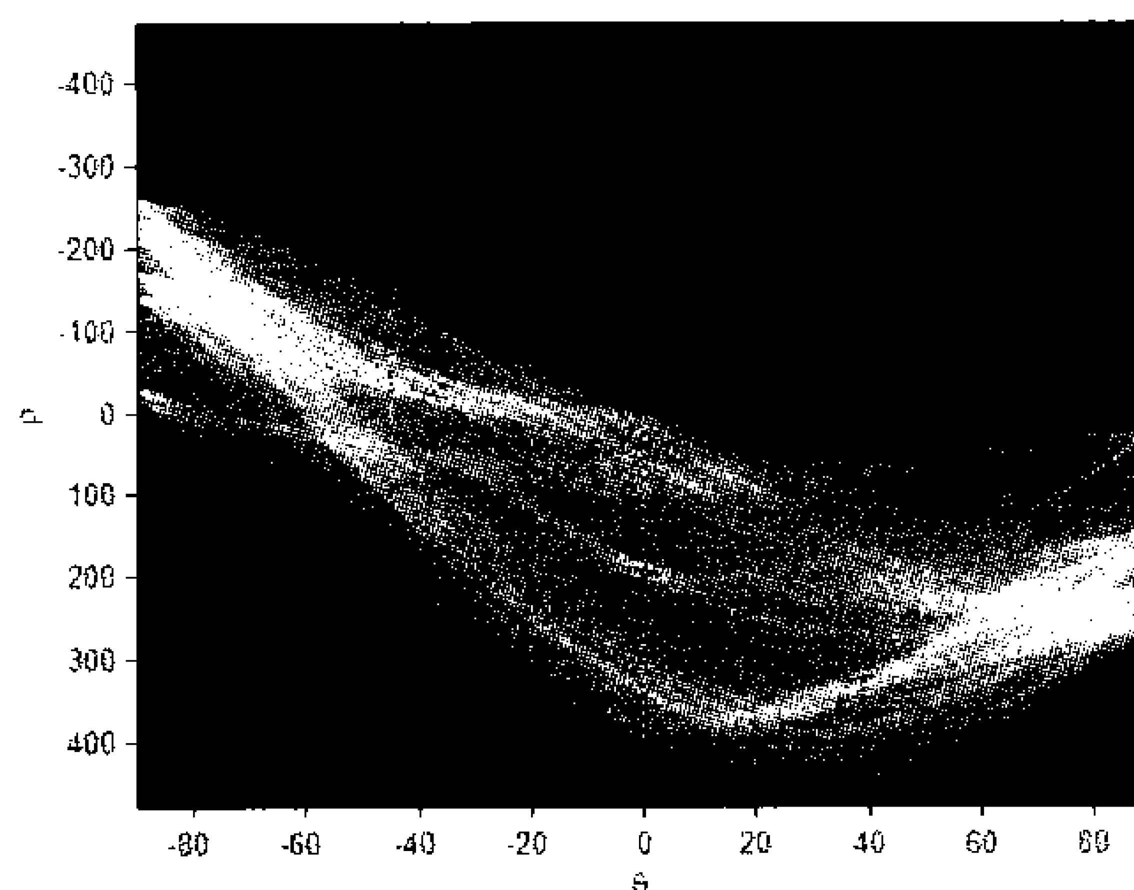


图 7-9 Hough 变换的结果

H 矩阵上明显有高叠加点，也就是说，图像中包含有大量的直线。利用阈值取舍，我们可得到其检测的最终结果，如图 7-10 所示。

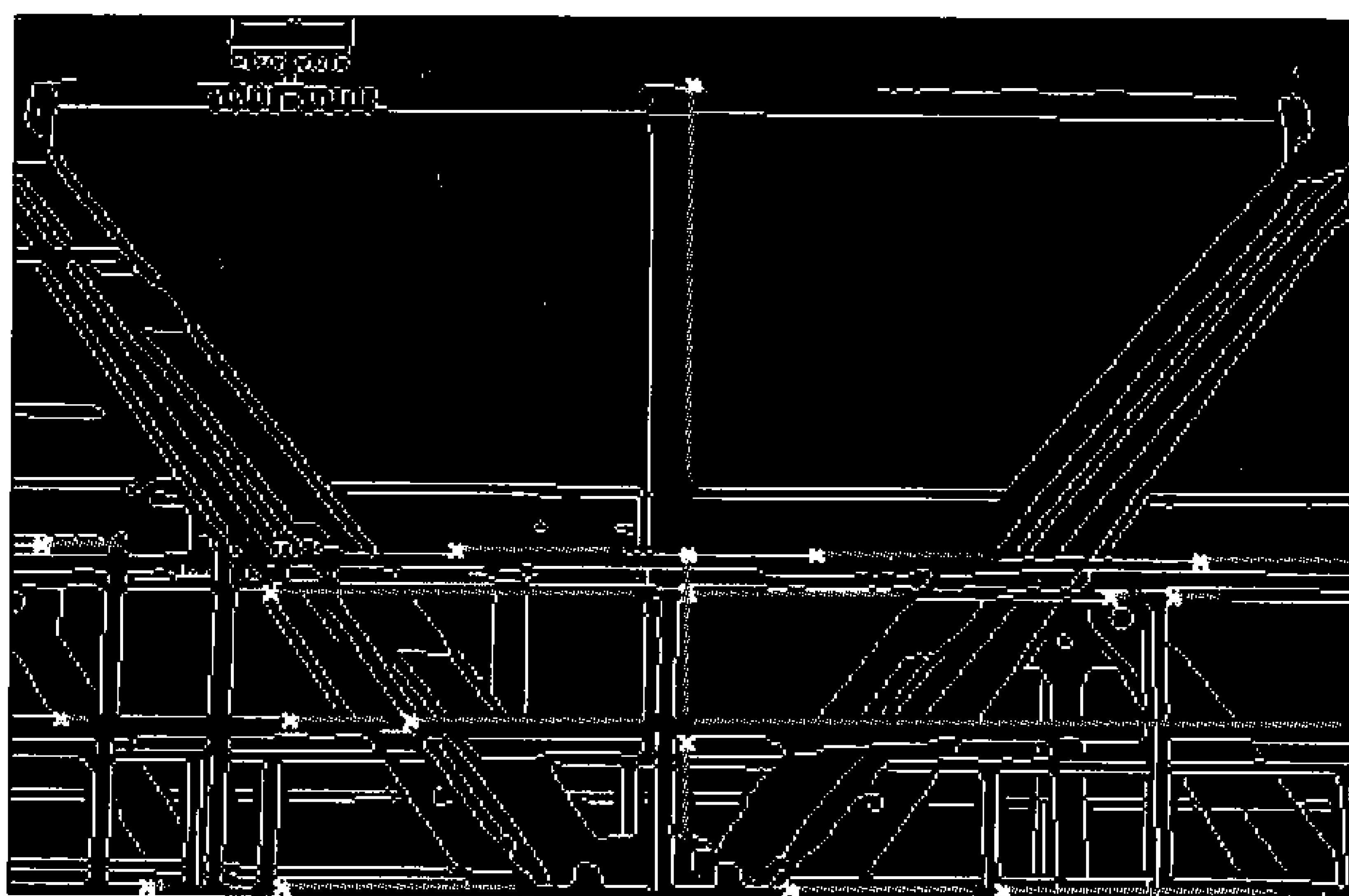


图 7-10　houghline 的检测结果

从图 7-10 可以看出，利用 Hough 可以有效的检测出图像的直线段。不过，为了得出较好的结果，我们必须多次调整检测的阈值。

7.2.2　MATLAB 实现图像的边缘检测

边缘检测在图像处理与计算机视觉中占有特殊的位置，它是底层视觉处理中最重要的环节之一，也是实现基于边界的图像分割的基础。在图像中，边界表明一个特征区域的终结和另一个特征区域的开始，边界所分开区域的内部特征或属性是一致的，而不同区域内部的特征或属性是不同的，边缘的检测正是利用物体和背景在某种图像特性上的差异来实现的。这种差异包括灰度、颜色或者纹理特征。边缘检测实际上就是检测图像特性发生变化的位置。

在 MATLAB 中，利用图像处理工具箱中的 edge 函数可以实现基于各种算子的检测边缘的功能，edge 函数调用格式如下：

```
[g,t]=edge(I, 'method', parameters)
```

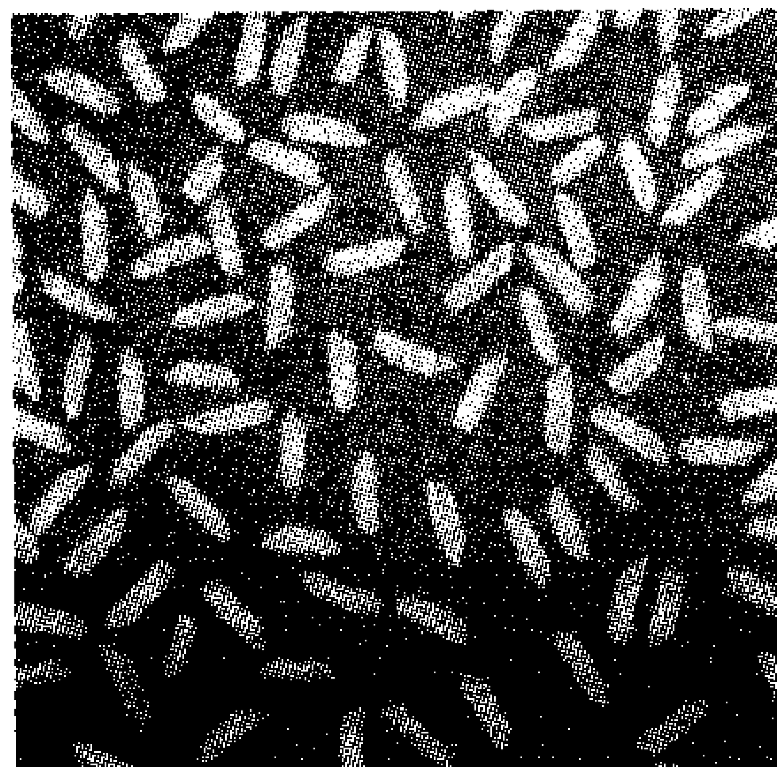
其中，*I* 是输入图像，method 是表 7-1 中列出的一种方法，parameters 是我们后面将要说明的另一个参数。在输出中，*g* 是一个逻辑数组，其值如下决定：在 I 中检测到边缘的位置为 1，在起他位置为 0。参数 *t* 是可选的,它给出 edge 使用的阈值，以确定哪个梯度值足够大到可以称为边缘点。

表 7-1　函数 edge 中可用的边缘检测器

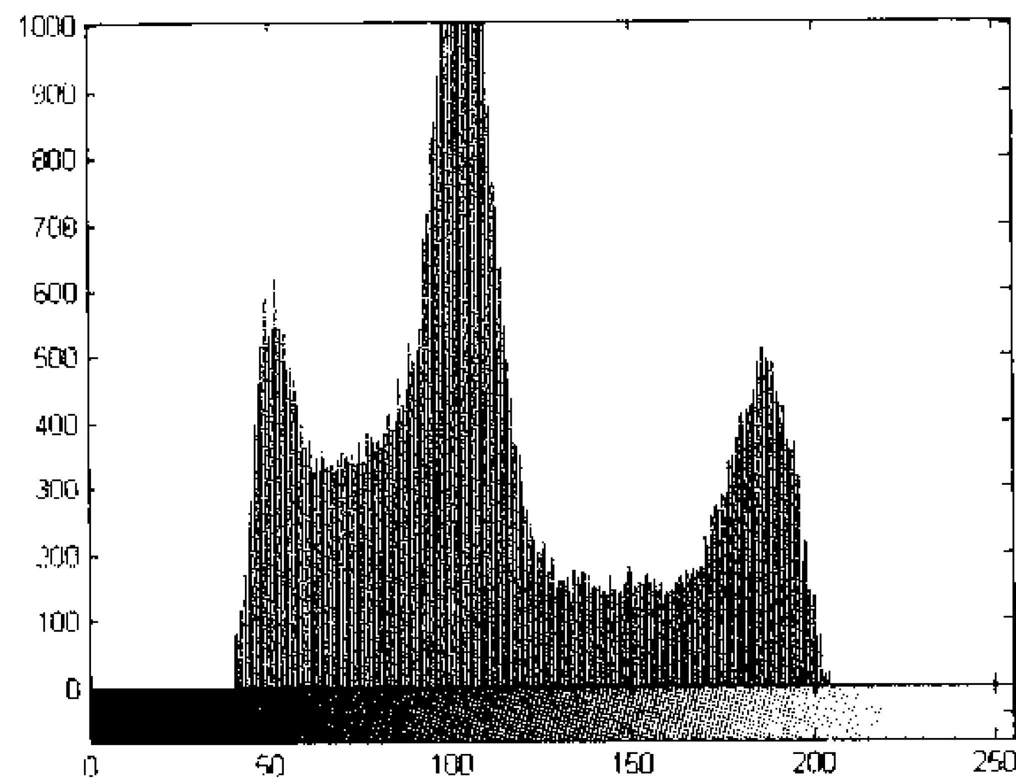
参　数		描　述
method	'roberts'	Robert 算子
	'sobel'	Sobel 算子
	'prewitt'	Prewitt 算子
	'log'	log 算子
	'zerocross'	零交叉方法
	'canny'	Canny 算子

例 7.8 以 sobel 算法进行边缘检测。

```
clc
clear all
image=imread('rice.png');
imhist(image)                                    %显示图像的直方图
image0=edge(image,'sobel');                      %自动选择阈值的 sobel 算法
image1=edge(image,'sobel',0.06);                 %指定阈值为 0.06
image2=edge(image,'sobel',0.04);                 %指定阈值为 0.04
image3=edge(image,'sobel',0.02);                 %指定阈值为 0.02
figure;imshow(image)
figure,
xlabel('原图')
subplot(221);imshow(image0)
xlabel('默认门限')
subplot(222);imshow(image1)
xlabel('门限 1')
subplot(223);imshow(image2)
xlabel('门限 2')
subplot(224);imshow(image3)
xlabel('门限 3')
```



原图



图像的直方图



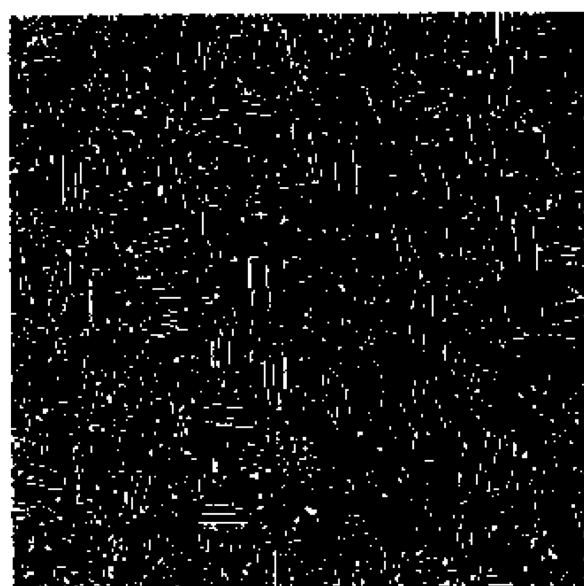
默认门限



门限1



门限2



门限3

图 7-11 例 7.8 运行结果

从本例可以看出，若要设定临界值，首先要看灰度直方图的分布，寻找其分界的地方为临界值，以本图像为例。差不多取到 0.06，其边缘效果就已经很明显了。

7.2.3 MATLAB 在汽车牌照识别系统中的应用

汽车牌照自动识别系统是以汽车牌照为特定目标的专用计算机视觉系统，是计算机视觉和图像模式识别技术在智能交通领域应用。车牌的自动识别系统主要包括车牌定位和车牌字符识别两部分。它可广泛应用于交通流量检测、交通控制与诱导、机场、港口、小区的车辆管理、不停车自动收费、闯红灯等违章车辆监控以及车辆安全防盗等领域，具有广阔的应用前景。但由于车牌识别要适应各种复杂背景以及不同光照条件影响，使车牌分割及识别增加了难度，目前虽然国内外都有一些实用的车牌识别系统面市，但这些系统的应用都存在一定的约束，至今车牌自动识别技术尚未达到很完善的程度，因此吸引了很多国内外学者的研究兴趣。本节分别从汽车牌照定位和字符识别从两方面，说明 MATLAB 在汽车牌照自动识别系统中的应用。

图 7-12（a）显示的是一副有牌照的车辆的原始图片。在对图像进行处理前先介绍一个非常有用的函数：pixval。它经常被用来交互地显示单个像素的亮度值。该函数可以显示覆盖在图像上的光标。当光标随着鼠标在图像上移动时，光标所在位置的坐标和该点的亮度值会在该图形窗口的下方显示出来。处理彩色图象时，红、绿、蓝分量的坐标也会显示出来。若按下鼠标左键不放，则 pixval 将显示光标初始位置和当前位置间的欧几里得距离。

在汽车牌照识别中，主要是将牌照部分突出显示出来进行字符识别，去除其他一切不相关因素。

车牌区域的识别是基于以下思想的：分析图像，使用 pixval 函数来获得牌照的背景色的红、绿、蓝分量亮度值和坐标；通过统计算法找出车牌的范围；然后通过修剪得到最终图像。本例所用程序如例程 7.9 所示。

例程 7.9　车牌识别

```
%%%%%%
clc
clear
I=imread('Car.jpg');
subplot(2,3,1),imshow(I);
title('原始图像')

%%%提取原始图象行、列、维数
[y,x,z]=size(I);
myI=double(I);

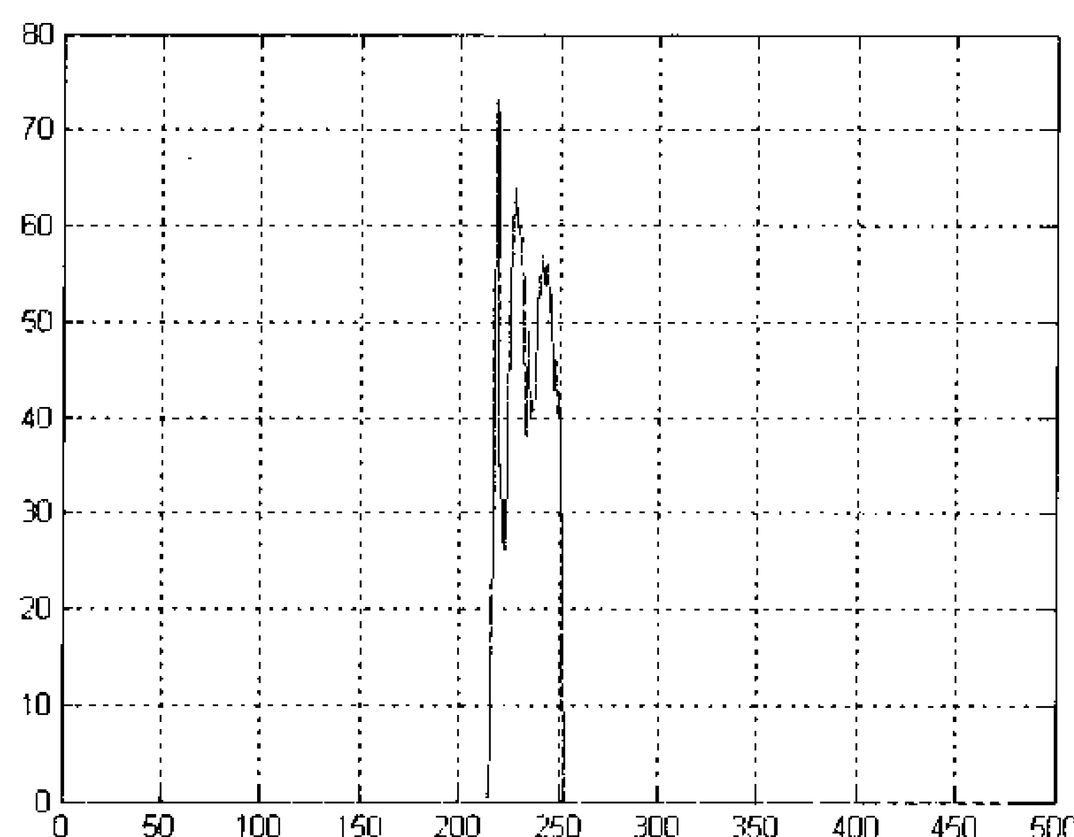
%%%Y 方向统计分析
Blue_y=zeros(y,1);
for i=1:y
    for j=1:x
            if((myI(i,j,1)<=121)&&myI(i,j,1)>=110&&((myI(i,j,2)<=155)&& ...
```

```
                (myI(i,j,2)>=141))&&((myI(i,j,3)<=240)&&(myI(i,j,3)>=210))) %%%蓝色 RGB
                                                                              的灰度范围
                Blue_y(i,1)= Blue_y(i,1)+1;        %%%蓝色象素点统计
            end
    end
end
[temp MaxY]=max(Blue_y);            %%%Y 方向车牌区域确定
PY1=MaxY;
while ((Blue_y(PY1,1)>=5)&&(PY1>1))
        PY1=PY1-1;
end
PY2=MaxY;
while ((Blue_y(PY2,1)>=5)&&(PY2<y))
        PY2=PY2+1;
end
IY=I(PY1:PY2,:,:);
subplot(2,3,2),plot(Blue_y);grid
title('Y 方向统计')
subplot(2,3,4),imshow(IY);
title('Y 方向车牌区域确定')
%%%X 方向统计分析
Blue_x=zeros(1,x); %%%进一步确定 X 方向的车牌区域
for j=1:x
    for i=PY1:PY2
            if((myI(i,j,1)<=121)&&myI(i,j,1)>=110&&((myI(i,j,2)<=155)&& ...
                (myI(i,j,2)>=141))&&((myI(i,j,3)<=240)&&(myI(i,j,3)>=210)))
                Blue_x(1,j)= Blue_x(1,j)+1;
            end
    end
end
PX1=1;
while ((Blue_x(1,PX1)<3)&&(PX1<x))
        PX1=PX1+1;
end
PX2=x;
while ((Blue_x(1,PX2)<3)&&(PX2>PX1))
        PX2=PX2-1;
end
subplot(2,3,3),plot(Blue_x);grid
title('X 方向统计')

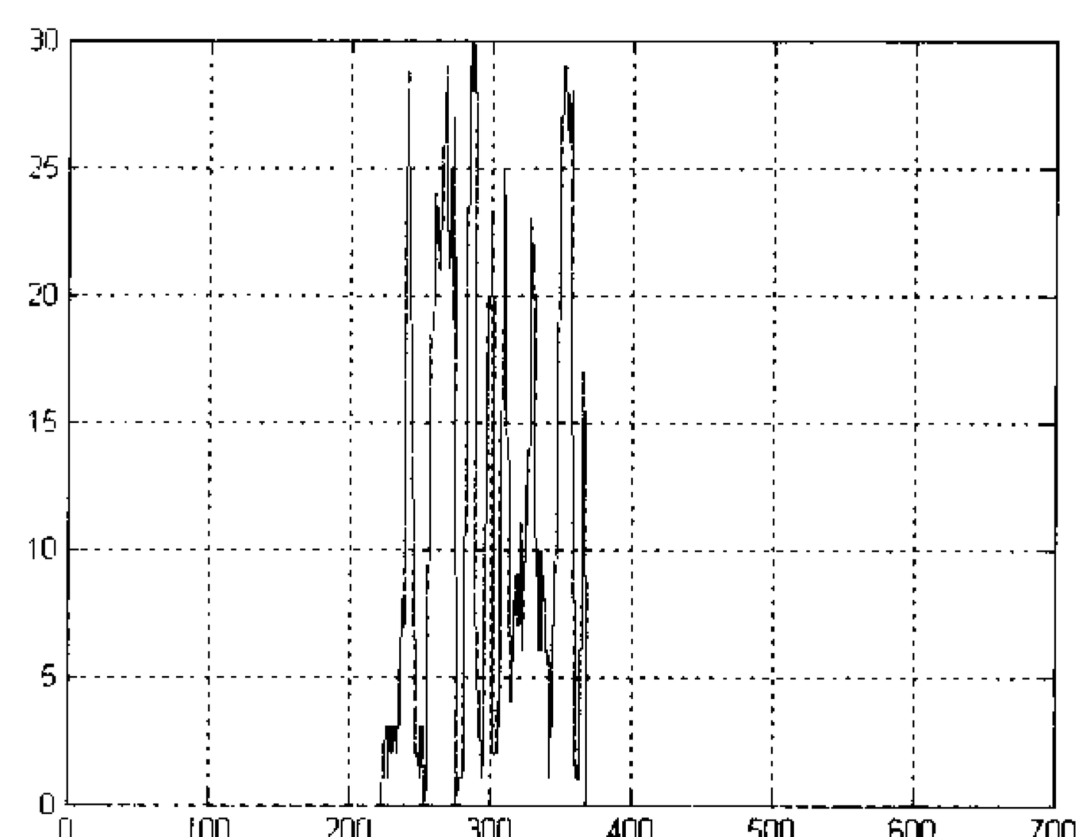
%%% 对车牌区域的修正
PX1=PX1-2;
PX2=PX2+2;
Plate=I(PY1:PY2,PX1-2:PX2,:);
subplot(2,3,5),imshow(Plate);
title('车牌显示')
```



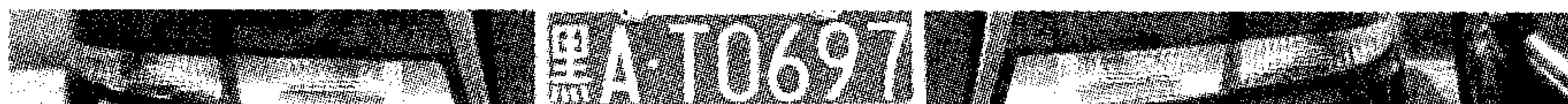
（a）原始图像



（b）*X* 方向统计



（c）*Y* 方向统计



（d）*Y* 方向车牌区域确定



（e）车牌显示

图 7-12　汽车牌照识别

通过分析图 7-12（a）可以看到，车体和车牌颜色对比明显，车体主要为黑色，而车牌主要以蓝色为背景，字体为白色。车体和车牌颜色的明显对比为本例的算法提供了基础。

先对牌照背景色在 *X*、*Y* 方向上统计，做出统计图如图 7-12（b）、（c）所示，分别表示 *X* 方向统计和 *Y* 方向统计。统计的目的主要是找到车牌背景在整个图像中的坐标范围。统计完成以后，实际上就得到了车牌的大体外围。再对原始图像进行切割，显示它在 *Y* 方向的图像范围，如图 7-12（d）所示，然后找出它在 *X* 方向的图像范围，最后对其进行修剪、放大、结果显示，如图 7-12（e）所示。

7.3　MATLAB 与控制工程

MATLAB R2007 控制产品支持控制设计过程中的每一个环节，从控制理论方法的设计仿真到面向不同领域的应用，如汽车、航空航天、计算机和通信等，MATLAB 控制系列产品均提供了强大的计算能力，主要表现在以下几个方面。

（1）使用 MATLAB 语言，只需要花很短的时间就可以开发出控制算法。而使用 MATLAB 强大的绘图能力可以方便地对数据、方程和结果进行显示，如绘制根轨迹、波特图、响应和谱等。

（2）与控制相关的工具箱涵盖了许多前沿的控制设计方法。MATLAB 提供的图形界面使控制系统的设计和分析变得更加方便。

（3）使用框图式动态系统仿真工具 Simulink，可以方便地建立控制系统原型和控制对象模型，通过仿真不断地优化和改善设计。无论是离散的、连续的、条件执行的、多采样的或混杂的系统，Simulink 都是描述动态系统模型的最佳工具。

（4）结合有限状态机、状态转移图和流程图等多种技术手段，Stateflow 使用户能够建立复杂响应系统的清晰、简明的描述。通过 Simulink 和 Stateflow，用户可以在单一的集成环境下对包括顶层控制逻辑、物理对象和控制器的整个系统进行建模与仿真。

（5）Real-Time Workshop 和 Stateflow Coder 可以直接从 Simulink 模型和 Stateflow 图中生成高质量的代码，并自动地进行编译、链接并下载可执行文件到目标处理器上。通过将代码生成工具与先进的实时系统集成，用户可以快速、方便地实现系统控制原型并实时地仿真和分析自己的设计。

控制系统工具箱主要用于反馈控制系统的分析、设计和仿真，所涉及的领域涵盖经典控制理论和现代控制理论的大部分内容，包括根轨迹、极点配置和线性二次最优控制器设计等。通过控制系统工具箱，用户可以创建线性时不变系统的传递函数、零极点—增益模型或状态空间模型。控制系统工具箱既适用于连续时间系统，也适用于离散时间系统，并且可以实现不同模型之间的相互转换。用户还能够轻松地绘制系统的时域或频域响应和开环系统的根轨迹图。其中的控制系统设计函数能够快速完成系统的极点配置、最优控制器设计等。MATLAB 自身提供的开放式环境可以让用户通过 M 文件建立自己的控制模型和控制算法。

除了控制系统工具箱之外，MathWorks 公司还在 MATLAB 软件包中集成了控制系统分析和设计的其他相关工具箱和软件包。这些工具箱有的是控制系统工具箱的补充，有的是独立的软件包，主要以控制系统为研究对象。如表 7-2 所示是 MATLAB 软件提供的控制工程设计相关的模块。

表 7-2　MATLAB 控制工程设计相关模块

产 品 模 块	含　义
MATLAB	用于数据分析，可视化与算法开发的高性能科学计算环境
MATLAB Toolboxes	一组面向专业领域的、经过高度优化了的 MATLAB 函数。这些领域包括：控制系统设计与分析、系统识别、信号处理、控制逻辑、神经网络等
Simulink	用于真实世界的非线性动态系统建模与设计的图形化仿真环境
Simulink Blocksets	Simulink 模块包含一组专业模块支持如下专业领域：电力系统建模、DSP 与定点算法开发
Stateflow	用于建模与设计复杂控制逻辑的图形化仿真环境
Stateflow Coder	从 Stateflow 流程图中直接生成 C 代码的工具
Real-Time Workshop	从 Simulink 模型中直接生成 C 或 Ada 代码的工具
Real-Time Windows Target	用于生成在 PC 上可交互运行 Simulink 模型实时原型代码的工具
xPC Target	用于生成在指定的目标 PC 上实时运行 Simulink 模型实时代码的快速原型与实施工具
MathWorks Partner Products	与 MathWorks 工具兼容的第三方软硬件产品。它们支持快速原型与实时测试用户基于 MATLAB 与 Simulink 的系统设计

借助这些工具箱和软件包，用户可以完成诸如系统辨识、系统建模、系统仿真及鲁棒控制、模糊控制和神经控制等系统设计任务。所包含的内容几乎涉及现代控制理论的所有内容。这些工具箱的简要说明如下。

1．系统辨识工具箱（System Identification Toolbox）

系统辨识工具箱基于预先测试得到的输入/输出数据来建立动态系统的线性模型。可以使用时域/频域技术对单通道数据或多通道数据进行模型辨识。利用该工具箱可以对那些不容易用数学方法描述的动态系统建立数学模型，包括发动机系统、飞行动力学系统及机电系统等。

2．模糊逻辑工具箱（Fuzzy Logic Toolbox）

模糊逻辑工具箱利用基于模糊逻辑的系统设计工具扩展了MATLAB的科学计算。通过GUI，可以完成模糊推理系统设计的全过程。工具箱中的函数提供了多种通用的模糊逻辑设计方法。通过该工具箱可以利用简单的模糊规则对复杂的系统行为进行建模，然后将这些规则应用于模糊推理系统。

3．鲁棒控制工具箱（Robot Control Toolbox）

鲁棒控制工具箱提供了用于高级“鲁棒”多变量反馈控制系统设计的专业工具，可以与MATLAB和控制系统工具箱一起使用。

4．LMI控制工具箱（LMI Control Toolbox）

LMI（Linear Matrix Inequality）控制工具箱，即线性矩阵不等式控制工具箱提供了一个通用的集成环境用来刻画和求解LMI问题，同时也提供了基于LMI进行控制系统设计的工具，其强有力的功能及友好的用户界面能帮助工程师开发自己特定的解决LMI问题的应用程序。

5．模型预估计控制工具箱（Model Predictive Control Toolbox）

模型预估计控制工具箱用于设计、分析和仿真基于MATLAB建立或Simulink线性化所得到的对象模型的模型预估计控制器。该工具箱提供了所有与模型预估计控制系统设计相关的主要特性。

6．μ分析及综合工具箱（μ-Analysis & Synthesis Toolbox）

μ分析及综合工具箱收纳了用于分析模型非确定性对闭环性能的影响和带有非确定性模型的最坏表现状况。工具箱提供了运用$H\infty$优化多输入多输出系统控制器的综合性工具。

大型控制系统的设计和开发是工程领域的一大难题，特别是其中复杂的设计、仿真和计算问题，MATLAB则提供了一整套解决方案。MathWorks公司通过提供一组完整的、紧密集成的工具，形成了支持整个工程设计流程的、无缝集成的控制系统设计解决方案。这一解决方案可以使用户方便地穿梭于建模、仿真、验证与实施之间而无须转换数据、重写代码或改变软件环境。如图7-13所示是MathWorks产品支持控制流程。

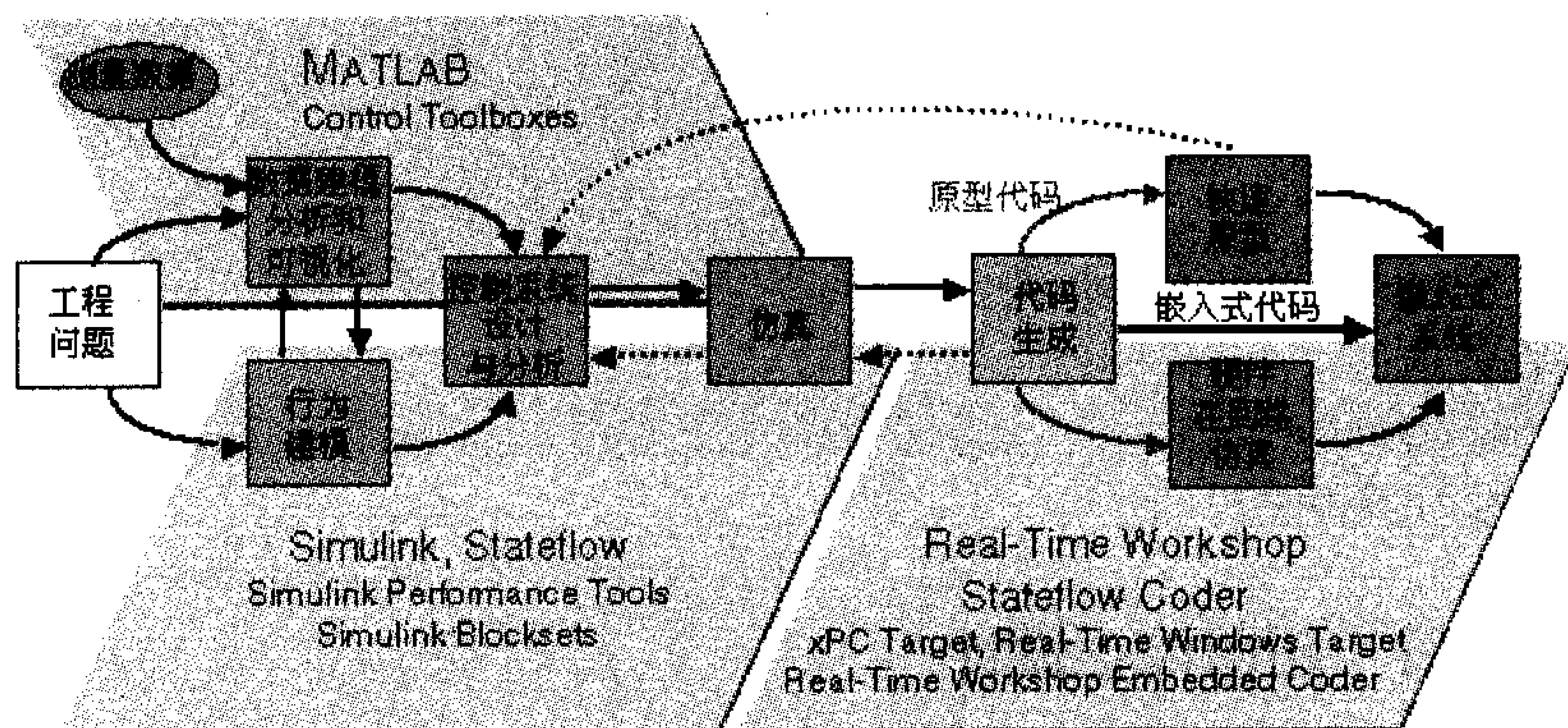


图 7-13　MathWorks 产品支持控制流程

7.3.1　控制系统建模与分析

例 7.10　现有如下的 SISO 系统，其传递函数为：

$$H(s)=\frac{3s+1}{2s^2+5s+11}$$

试创建该系统的传递函数模型。

解：调用 MATLAB 函数命令 tf()执行如下的 MATLAB 程序：

```
num=[3 1];
den=[2 5 11];
h=tf(num,den)
```

程序运行后即得到该系统的传递函数模型如下：

```
Transfer function:
    3 s + 1
------------------
2 s^2 + 5 s + 11
```

例 7.11　已知离散系统方程如下：

$$\begin{cases} x(k+1)=fx(k)+gu(k) \\ y(k)=cx(k)+du(k) \end{cases}$$

假设方程中有：

$$f=[0.8760\quad 0\quad 0;\quad 0.2546\quad 0.6621\quad -0.5701;\quad 0.1508\quad 0.4221\quad 1];$$

$$g=[0.2105;\quad 0.1033;\quad 0.1768];c=[0\quad 1\quad 3.5];d=0;$$

采样周期 T=0.1s，试确定离散系统的可控性。

解：根据状态完全可控的充分必要条件来分析系统的可控性。执行下面的 MATLAB 程序，求取系统的可控性矩阵：

```
f=[0.8760 0 0;0.2546 0.6621 -0.5701;0.1508 0.4221 1];
g=[0.2105;0.1033;0.1768];
c=[0 1 3.5];
d=0;
n=3;
CAM=ctrb(f,g);
```

```
    rcam=rank(CAM);
    if rcam==n
        disp('System is controlled')
    elseif rcam<n
        disp('System is no controlled')
```

程序运行结果是：

```
    System is controlled
```

即表示该系统完全可控。

7.3.2 波特图滞后 超前校正设计

基于波特图进行相位滞后－超前校正的设计步骤如下。

（1）根据要求的稳态品质指标，确定系统开环增益 K 值。

（2）根据求得的 K 值，画出校正前系统的波特图，并检验性能指标是否满足要求。

（3）确定滞后校正器传递函数的参数 $G_{c1}(s)=\dfrac{1+T_1 s}{1+\beta T_1 s}$，式中，$\beta>1,\dfrac{1}{T_1}<\omega_{c1},\dfrac{1}{\beta T_1}<\omega_{c1}$，$\dfrac{1}{T_1}$ 要距 ω_{c1} 较远为好。工程上常选择：$\dfrac{1}{T_1}=0.1\omega_{c1}$，$\beta=8\sim10$。

（4）选择一个新的系统剪切频率 ω_{c2}，使在这一点上超前校正器所提供的相位超前量达到系统相位稳定裕量的要求，并使在这一点上原系统加上滞后校正器的综合幅频特性衰减为 0dB，即 L 曲线在 ω_{c2} 点穿越横坐标。

（5）确定超前校正器传递函数的参数 $G_{c2}(s)=\dfrac{1+T_2 s}{1+\alpha T_2 s}$，式中 $a<1$。

由表达式：$20\log\alpha=L(\omega_{c2})$（$L$ 为原系统加上滞后校正器后幅频分贝值），可得：

$$\omega_{cn}=\omega_m=\frac{1}{\sqrt{\alpha}T}\text{，}\quad T=\frac{1}{\sqrt{\alpha}\omega_m}$$

求出参数 a、T。

（6）校验系统性能指标。

例 7.12 设单位反馈系统的开环传递函数为：$G(s)=\dfrac{K_0}{s(s+1)(s+4)}$，试用波特图设计法设计滞后－超前校正装置，使校正后系统满足如下性能指标。

- 在单位斜坡信号作用下，系统的速度误差系数 $K_v=10s^{-1}$。
- 系统校正后剪切频率 $\omega_c\geqslant1.5s^{-1}$。
- 系统校正后相角稳定裕度 $\gamma\geqslant40^\circ$。
- 校正后系统时域性能指标：$\sigma\%\leqslant30\%,t_p\leqslant2s,t_s\leqslant6s$。

解：

（1）求 K_0。

根据自动控制理论，单位斜坡响应的速度误差系数 $K_v = K = 10s^{-1}$。根据速度误差的定义 $K_v = \lim_{s \to 0} sg \dfrac{K_0}{s(s+1)(s+4)} = 10$，可得 $K_0 = 40s^{-1}$。

被控对象的传递函数为：

$$G(s) = \frac{40}{s(s+1)(s+4)}$$

（2）作原系统的波特图与阶跃响应曲线。

在 MATLAB 命令栏中输入：

```
>> k0=30;
n1=1;d1=conv(conv([1 0],[0.1 1]),[0.2 1]);
[mag,phase,w]=bode(k0*n1,d1);
figure(1);
margin(mag,phase,w);hold on
figure(2);
s1=tf(k0*n1,d1);
>> sys=feedback(s1,1);
>> step(sys)
```

可以得到如图 7-14 和图 7-15 所示的结果。

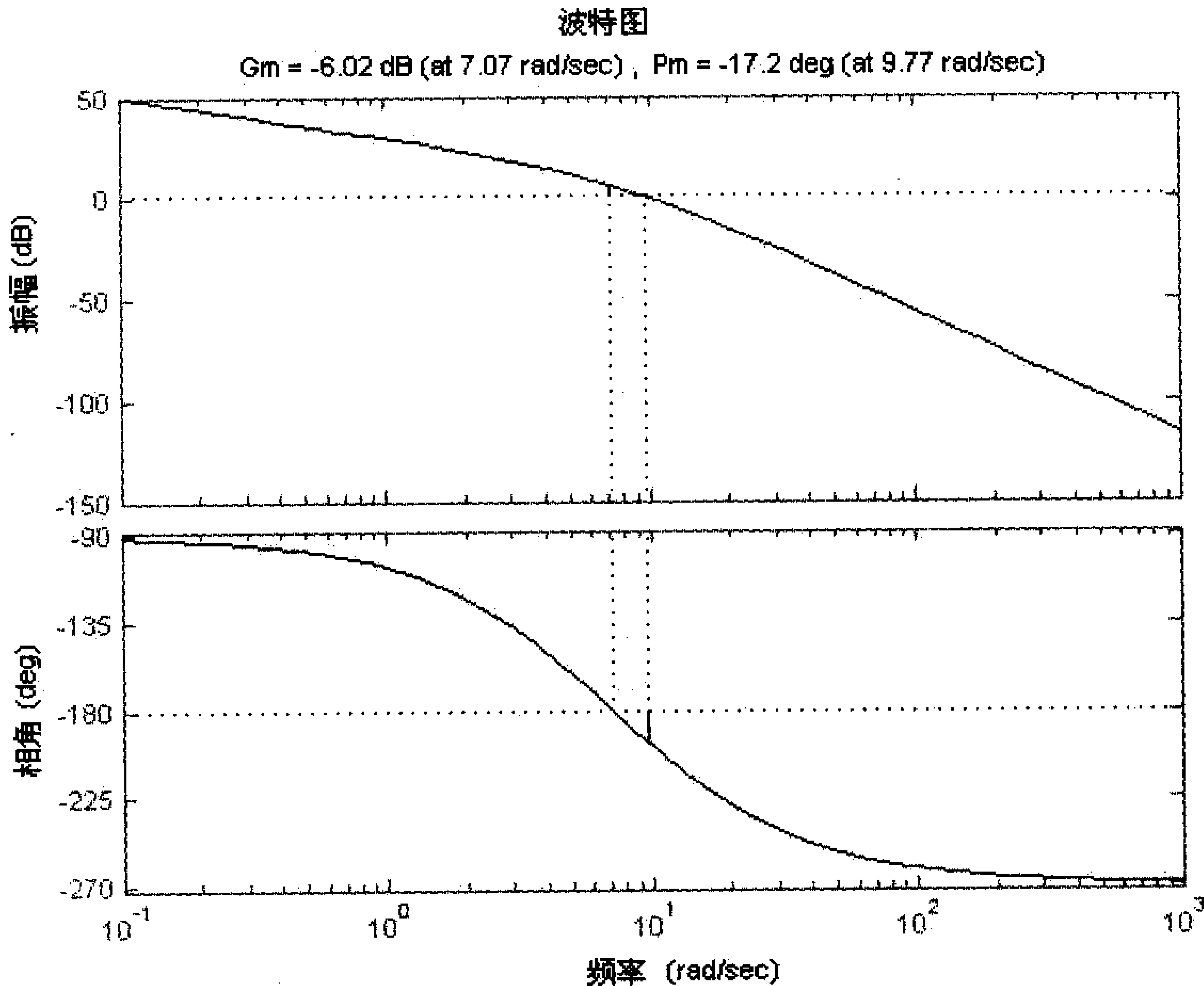


图 7-14　未校正系统波特图

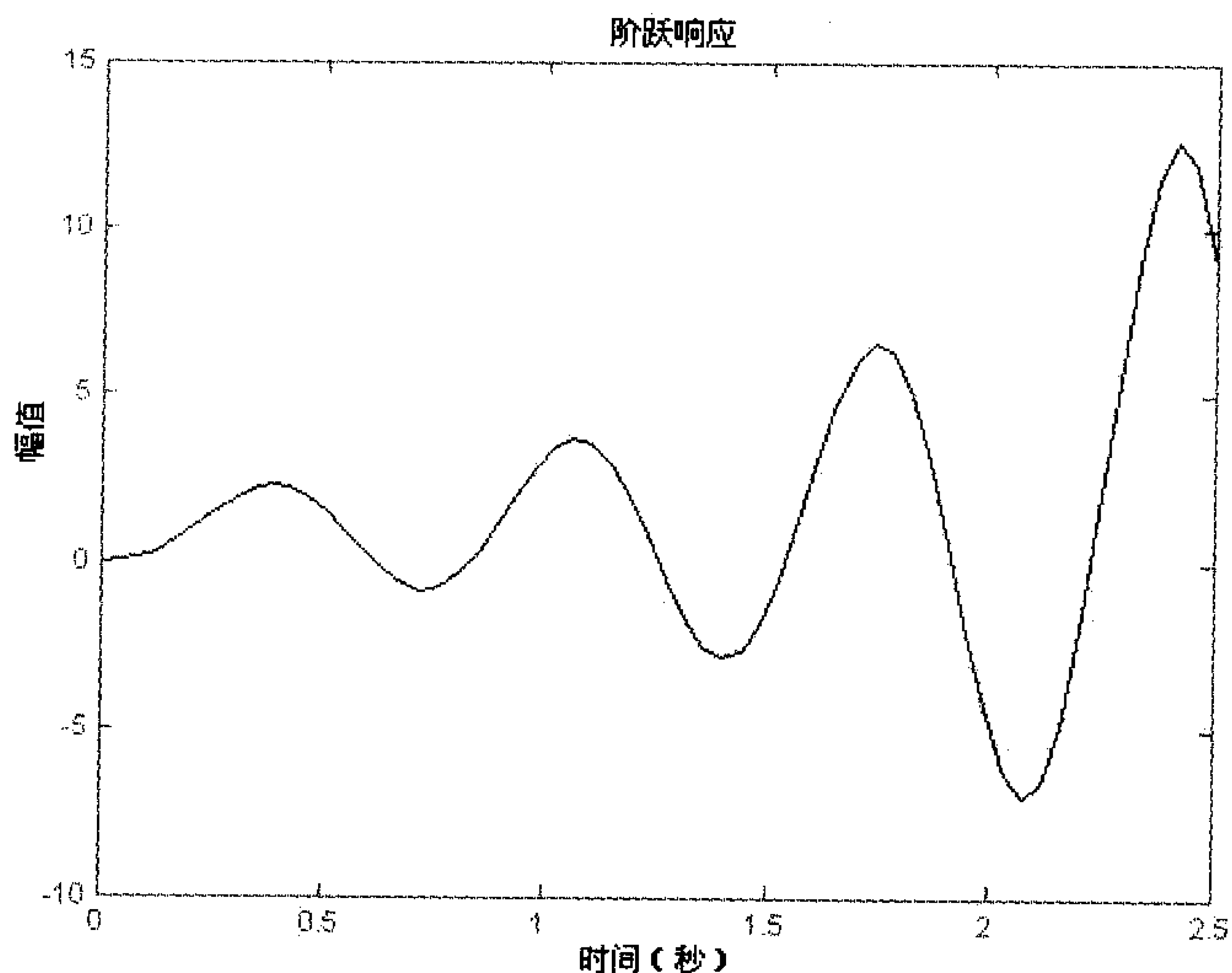


图 7-15　未校正系统阶跃响应曲线

由图 7-14 所示可以得到未校正系统的频域性能：

模稳定裕量：$G_m = -5.98\text{dB}$；　　　　$-\pi$ 穿越频率：$\omega_{cg} = 2\text{rad/sec}$

相稳定裕量：$P_m = -15\deg$；　　　　剪切频率：ω_{cp}=2.78rad/sec

由于系统的稳定裕量均为负值，此系统无法工作。此外，阶跃响应曲线发散，系统必须进行修正。

（3）求滞后校正器的传递函数。

根据题目要求，取校正后系统的剪切频率 $\omega_c = 1.5s^{-1}$，$\beta = 9.5$。根据滞后校正原理，给出程序如下：

```
% 求滞后校正器的传递函数
wc=1.5;k0=40;n1=1;
d1=conv(conv([1 0],[1 1]),[1 4]);
beta=9.5;
T=1/(0.1*wc);
betat=beta*T;
Gc1=tf([T 1],[betat 1])
```

运行程序后得到：

```
Transfer function:
6.667 s + 1
-----------
63.33 s + 1
```

即滞后校正器传递函数：

$$G_{c1}(s) = \frac{6.667s+1}{63.33s+1}$$

（4）求超前校正器的传递函数。

串联滞后校正器的系统传递函数为：$G_0(s)G_{c1}(s) = \dfrac{40}{s(s+1)(s+4)} g \dfrac{6.667s+1}{63.33s+1}$

给出求超前校正器传递函数的 MATLAB 程序如下：

```
% 求超前校正器的传递函数
n1=conv([0 40],[6.667 1]);
d1=conv(conv(conv([1 0],[1 1]),[1 4]),[63.33 1]);
sope=tf(n1,d1);
wc=1.5;
num=sope.num{1};
den=sope.den{1};
na=polyval(num,j*wc);
da=polyval(den,j*wc);
g=na/da;
g1=abs(g);
h=20*log10(g1);
a=10^(h/10);
wm=wc;
T=1/(wm*(a)^(1/2));
alphat=a*T;
Gc=tf([T 1],[alphat 1])
```

运行程序后得到：

```
Transfer function:
1.82 s + 1
-------------
0.2442 s + 1
```

超前校正器的传递函数为：

$$G_{c2}(s)=\frac{1.82s+1}{0.2442s+1}$$

（5）校验系统频域性能。

包含滞后－超前校正器的系统传递函数为：

$$G_0(s)G_{c1}(s)G_{c2}(s)=\frac{40}{s(s+1)(s+4)}g\frac{6.67s+1}{63.33s+1}g\frac{1.82s+1}{0.2442s+1}$$

给出如下程序：

```
% 校验
n1=40;d1=conv(conv([1 0],[1 1]),[1 4]);
s1=tf(n1,d1);
s2=tf([6.667 1],[63.33 1]);
s3=tf([1.82 1],[0.2442 1]);
sope=s1*s2*s3;
[mag,phase,w]=bode(sope);
margin(mag,phase,w)
```

程序运行后，得到校正后的系统波特图，如图 7-16 所示。

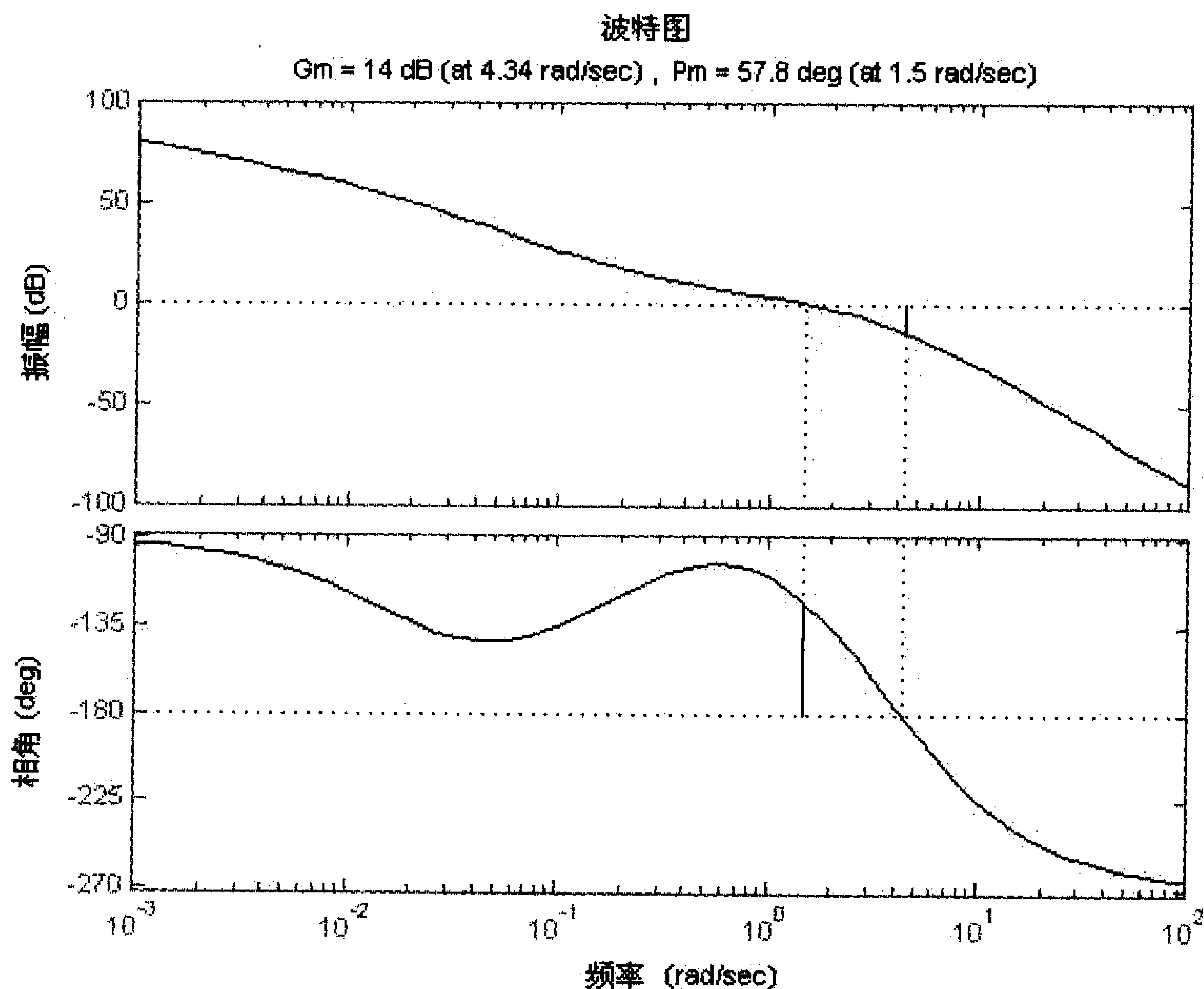


图 7-16 校正后系统的波特图

由图 7-16 可知，校正后系统的频域性能指标为：

模稳定裕量：$G_m = 14\text{dB}$；　　　$-\pi$ 穿越频率：$\omega_{cg} = 4.34\text{rad/sec}$

相稳定裕量：$P_m = 57.8\deg$；　　　剪切频率：$\omega_{cp} = 1.5\text{rad/sec}$

已满足题目要求。

（6）计算系统校正后阶跃响应曲线及性能指标。

用 MATLAB 语言编写如下的程序：

```
%校验后性能指标及阶跃响应
global y t;
k0=30;n1=40;d1=conv(conv([1 0],[1 1]),[1 4]);
s1=tf(n1,d1);
s2=tf([6.667 1],[63.33 1]);
s3=tf([1.82 1],[0.2442 1]);
sope=s1*s2*s3;
sys=feedback(sope,1);
step(sys)    %绘制阶跃响应曲线
[y,t]=step(sys);    %求出阶跃响应的函数值及其对应时间
[sigma,tp,ts]=ste(y,t)   %调用函数 ste()
```

程序运行后，得到校正后系统的单位阶跃响应曲线，如图 7-17 所示。

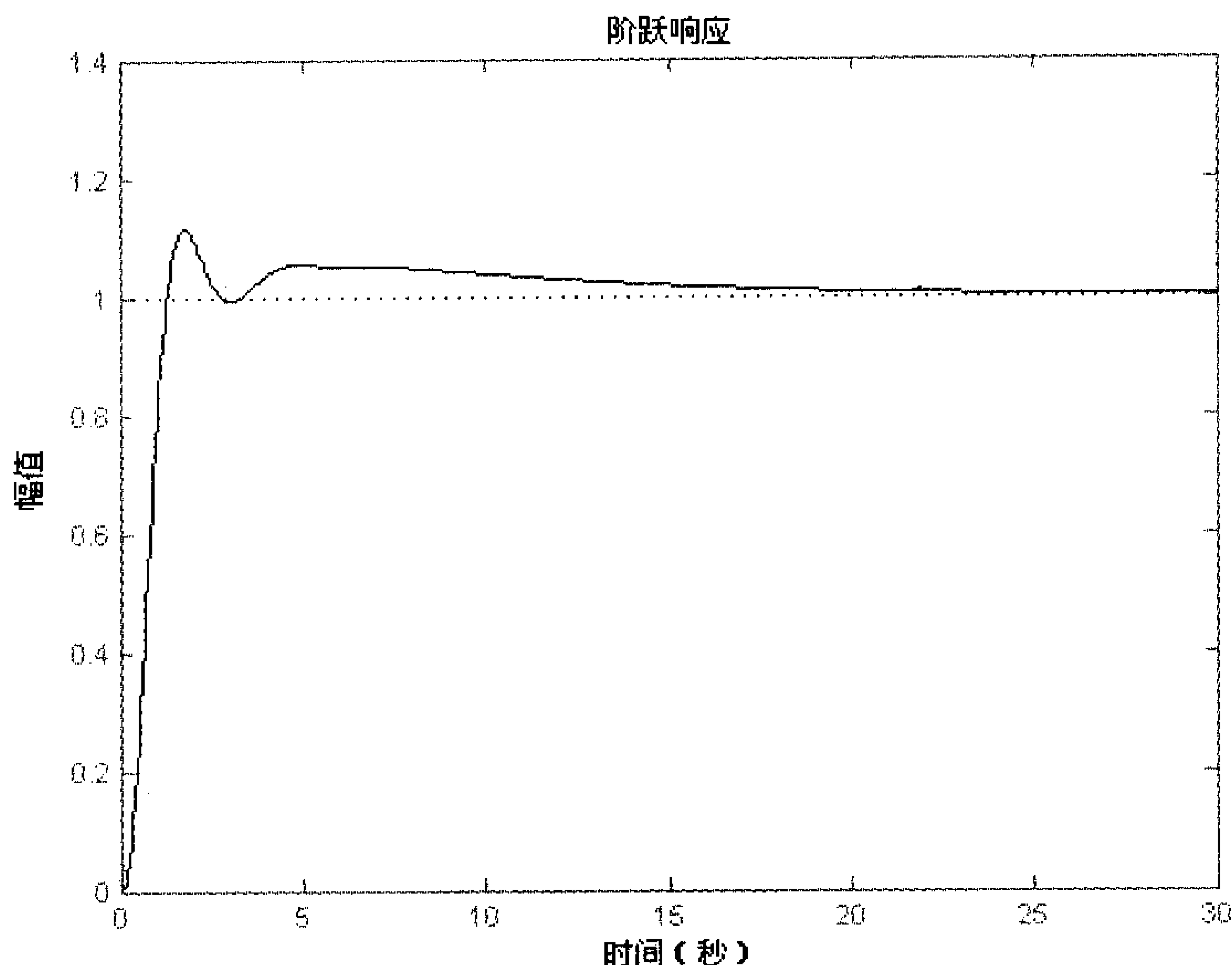


图 7-17 校正后系统的阶跃响应曲线

以及系统的阶跃响应性能指标：

```
sigma = 0.1144
tp = 1.7719
ts =5.6700
```

满足既定要求，时域性能指标合格。

7.3.3 PID 控制器设计

比例、积分、微分（PID）是建立在经典控制理论基础上的一种控制策略。PID 控制器作为最早实用化的控制器已有五十多年历史，现在仍然是最广泛的工业控制器。PID 控制器简单易懂，使用中不需要精确的系统模型等先决条件，因而成为应用最广泛的控制器。

传统 PID 控制的经验公式是 Ziegler 与 Nichols 在 20 世纪 40 年代初提出的。这个经验公式是基于带有延迟的一阶传递函数模型提出的。该对象模型可以表示如下：

$$G(s)=\frac{k}{1+sT}\mathrm{e}^{-sL}$$

在实际的过程控制系统中，有大量的对象模型可以近似地由这样的一阶模型来表示，如果不能物理地建立起系统的模型，我们还可以由实验提取相应的模型参数。如果实验数据是通过频域响应获得的，则我们可以容易地得出剪切频率 ω_c 和极限增益 K_c，设 $T_c=2\pi/\omega_c$，则 PID 控制器的参数也可以由表 7-3 给出。

表 7-3 Ziegler-Nichols 整定参数

控制器类型	由阶跃响应整定			由频域响应整定		
	K	T	T	K	T	T
P	T/kL			$0.5K$		
PI	$0.9T/kL$	$3L$		$0.4K$	$0.8T$	
PID	$1.2T/kL$	$2L$	$L/2$	$0.6K$	$0.5T$	$0.12T$

这里，我们来编写一个 MATLAB 函数 ziegler()，该函数的功能是实现由 Ziegler-Nichols 公式设计 PID 控制器，在今后我们设计 PID 控制器的过程中可以直接调用。给出程序如下：

```
function [Gc,Kp,Ti,Td,H]=ziegler(key,vars)
Ti=[ ]; Td=[ ]; H=[ ];
if length(vars)==4,
    K=vars(1); L=vars(2);
    T=vars(3); N=vars(4); a=K*L/T;
    if key==1,   Kp=1/a;       %P 控制器
    elseif key==2, Kp=0.9/a; Ti=3.33*L;    %PI 控制器
    elseif key==3      %PID 控制器
        Kp=1.2/a; Ti=2*L; Td=L/2;
    end
elseif length(vars)==3,
    K=vars(1); Tc=vars(2); N=vars(3);
    if key==1, Kp=0.5*K;
    elseif key==2, Kp=0.4*K; Ti=0.8*Tc;
    elseif key==3
        Kp=0.6*K; Ti=0.5*Tc; Td=0.12*Tc;
    end
elseif length(vars)==5,
    K=vars(1); Tc=vars(2);
    rb=vars(3); pb=pi*vars(4)/180;
    N=vars(5); Kp=K*rb*cos(pb);
    if key==2,
        Ti=-Tc/(2*pi*tan(pb));
    elseif key==3
        Ti=Tc*(1+sin(pb))/(pi*cos(pb));
        Td=Ti/4;
    end
end
switch key
    case 1, Gc=Kp;
    case 2, Gc=tf(Kp*[Ti,1],[Ti,0]);
    case 3
        nn=[Kp*Ti*Td*(N+1)/N, Kp*(Ti+Td/N), Kp];
        dd=Ti*[Td/N,1,0]; Gc=tf(nn,dd);
end
```

该函数的调用格式为：

$$\left[G_c, K_p, T_i, T_d\right] = \text{ziegler}(key, \text{var}\, s)$$

其中，*key* 为选择控制器类型的变量：当 *key*=1，2，3 时分别表示设计 P、PI、PID 控制器；若给出的是阶跃响应数据，则变量 $\text{var}\, s = [K, L, T, N]$；若给出的是频域响应数据，则变量 $\text{var}\, s = [K_c, T_c, N]$。

例 7.13　已知过程控制系统的被控对象为一个带延迟的惯性环节，其传递函数为：$G(s) = \dfrac{8}{360s+1} e^{-180s}$，试用 Ziegler-Nichols 法设计 P 控制器、PI 控制器和 PID 控制器。

解：由系统传递函数可得：*k*=80，*T*=360，*L*=180。

根据题意，利用 ziegler()函数计算系统 P、PI、PID 控制器的参数，并给出校正后系统阶跃响应曲线。程序如下：

```
% 设计 P、PI、PID 控制器
%
k=8;T=360;L=180;
n1=[k];d1=[T 1];G1=tf(n1,d1);
[np,dp]=pade(L,2);Gp=tf(np,dp);
[Gc1,Kp1]=ziegler(1,[k,L,T,1]);Gc1
[Gc2,Kp2,Ti2]=ziegler(2,[k,L,T,1]);Gc2
[Gc3,Kp3,Ti3,Td3]=ziegler(3,[k,L,T,1]);Gc3
G_c1=feedback(G1*Gc1,Gp);step(G_c1);hold on
G_c2=feedback(G1*Gc2,Gp);step(G_c2);
G_c3=feedback(G1*Gc3,Gp);step(G_c3);
```

运行该程序得到：

```
Gc1 =0.2500

Gc2=
134.9 s + 0.225
---------------
     599.4 s

Gc3=
19440 s^2 + 135 s + 0.3
-----------------------
   32400 s^2 + 360 s
```

以及经 P、PI、PID 校正后系统阶跃响应曲线，如图 7-18 所示。

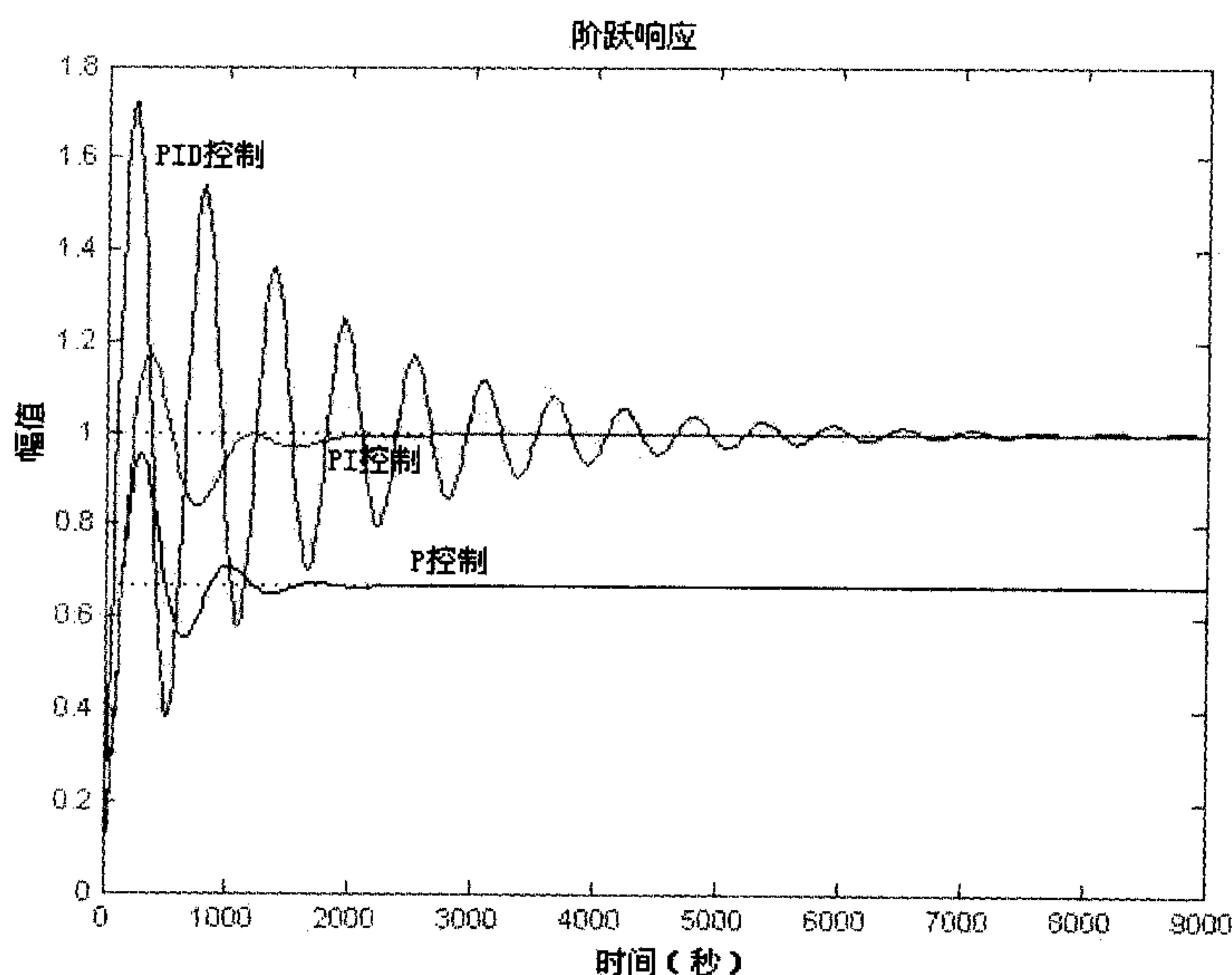


图 7-18　校正后系统阶跃响应曲线

由图 7-18 可知，用 Ziegler-Nichols 公式计算 P、PI、PID 校正器对系统校正后，其阶跃响应曲线中的 P、PI 校正二者响应速度基本相同，超调量终了值不同。PI 校正超调量比 P 校正的要小一些。PID 校正比前两者的响应速度都快，但超调量最大。

7.3.4　Kalman 滤波器

在实际应用中，若系统存在随机扰动，通常系统的状态需要由状态方程 Kalman 滤波器的形式给出。Kalman 滤波器就是最优观测器，能够抑制或滤掉噪声对系统的干扰和影响。利用 Kalman 滤波器对系统进行最优控制是非常有效的。

在 MATLAB 的工具箱中提供了 kalman()函数来求解系统的 Kalman 滤波器。其调用格式如下：

$$[kest, L, P] = \text{kalman}(sys, Q, R, N)$$

对于一个给定系统 *sys*，噪声协方差 Q，R，N 函数返回一个 Kalman 观测器的状态空间模型 *kest*，滤波器反馈增益为 L，状态估计误差的协方差为 P。用 MATLAB 构建的 Kalman 状态观测器模型为：

$$x(t) = \begin{pmatrix} A & LC \end{pmatrix}\hat{x}(t) + \begin{pmatrix} B & LD \end{pmatrix}u(t) + Ly(t)$$

$$\begin{pmatrix} \hat{y}(t) \\ \hat{x}(t) \end{pmatrix} = \begin{pmatrix} C \\ I \end{pmatrix}\hat{x}(t) + \begin{pmatrix} D \\ 0 \end{pmatrix}u(t)$$

例 7.14　已知系统的状态方程为：

$$x = \begin{pmatrix} -1 & 0 & 1 \\ 1 & 0 & 0 \\ -4 & 9 & -2 \end{pmatrix} x + \begin{pmatrix} 6 \\ 1 \\ 1 \end{pmatrix} u + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \omega$$

$$y = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} x + \upsilon$$

已知 $\omega(t)=10^{-3}, \upsilon(t)=0.1$，试设计系统 Kalman 滤波器。

解：为计算系统 Kalman 滤波器的增益矩阵与估计误差的协方差，给出以下程序 example8_9.m：

```
% 设计 Kalman 滤波器
A=[-1,0,1;1,0,0;-4,9,-2];
B=[6,1,1]';C=[0,0,1];D=0;
S=ss(A,B,C,D);
Q=0.001;R=0.1;
[kest,L,P]=kalman(S,Q,R);
L,P
```

运行程序，得到系统 Kalman 滤波器的增益矩阵 $\boldsymbol{L}$ 与估计误差的协方差 P 如下：

```
L =
    1.0641
    1.1566
    2.0393
P =
    0.0678    0.0664    0.1064
    0.0664    0.0695    0.1157
    0.1064    0.1157    0.2039
```

7.3.5 基于 LQR 的直升机飞行控制系统设计

1. 直升机飞行控制系统的动力学模型

直升机主要由机体、旋翼和尾桨等部分组成。机体又包括机身、垂直安定面及水平安定面，而机体的质心是指包含旋翼和尾桨质量的整个直升机的质心，垂直安定面和水平安定面合称为升力面。其机体坐标系如图 7-19 所示。

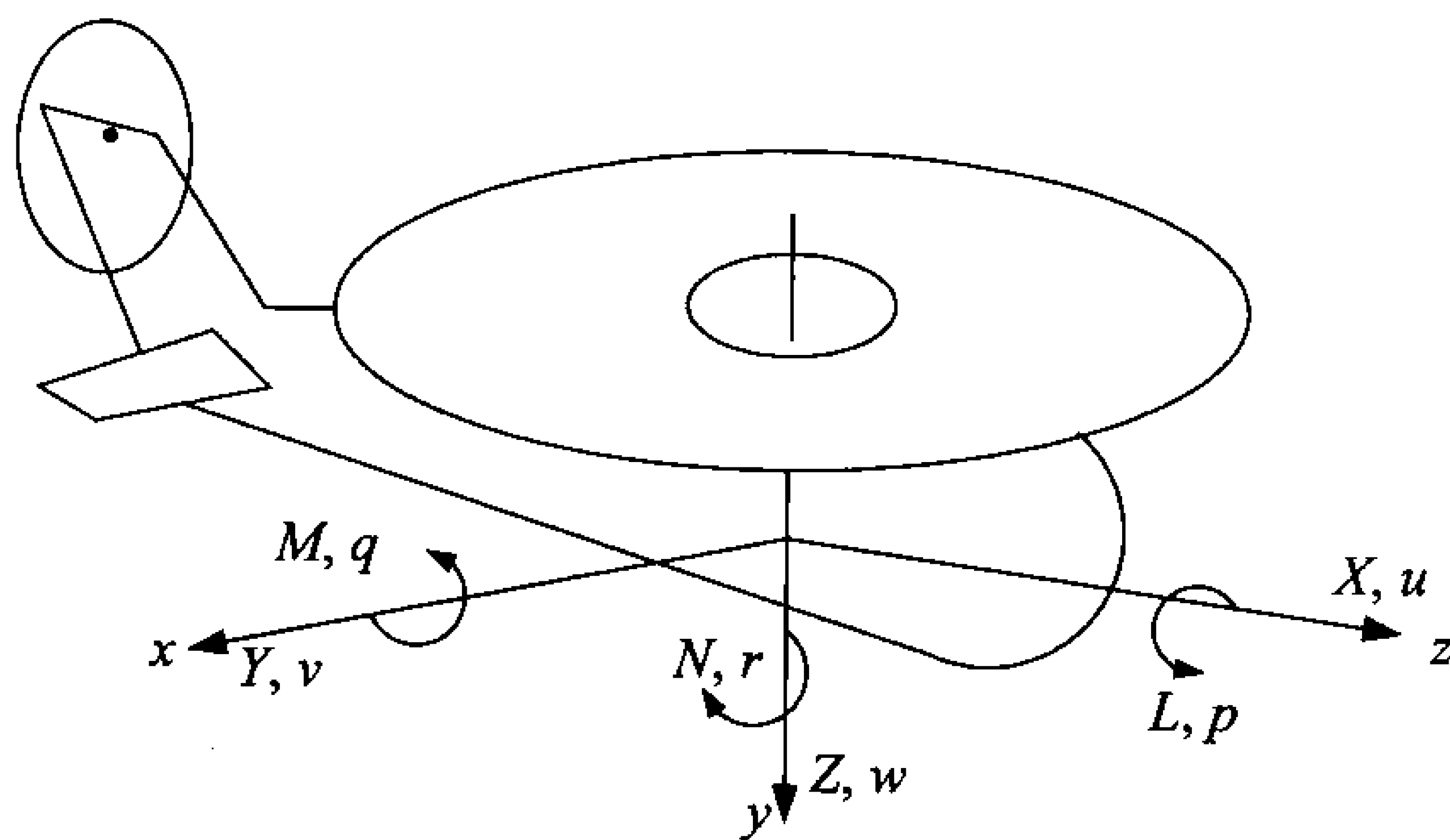


图 7-19 直升机机体坐标系

直升机机体作为一个六自由度（6doF）的刚体，其动力学方程和运动学方程与固定翼飞机的相同，可根据牛顿定律在机体坐标系中建立直升机机体的运动方程。假设符号 M_a 为直升机全机质量，X、Y、Z 为直升机机体轴 x、y、z 方向的外空气动力，L、M、N 为 x、y、z 轴的外空气动力矩，Ω_a 为主发动机转速，u、v、w 为机体坐标系 x、y、z 轴方向的速度；p、q、r 为机体坐标系 x、y、z 轴方向的角速度；θ 、ϕ 、Ψ 为相对于地轴系的俯仰角、

滚转角和偏航角。直升机的 6doF 非线性动力学方程由以下形式给出：

（1）机体质心平动方程：

$$u = (rv - qw) + \frac{X}{M_a} - g\sin\theta \tag{7.1}$$

$$v = (pw - ru) + \frac{Y}{M_a} + g\sin\phi\cos\theta \tag{7.2}$$

$$w = (qu - pv) + \frac{Z}{M_a} + g\cos\phi\cos\theta \tag{7.3}$$

（2）机体绕质心转动方程：

$$I_{xx}p = (I_{yy} - I_{zz})qr + I_{xz}(r + pq) + L \tag{7.4}$$

$$I_{yy}q = (I_{zz} - I_{xx})rp + I_{xz}(r^2 - p^2) + M \tag{7.5}$$

$$I_{zz}r = (I_{xx} - I_{yy})pq - I_{xz}(p - qr) + N \tag{7.6}$$

（3）姿态角与角速度的几何关系：

$$\psi = q\sin\phi\sec\theta + r\cos\phi\sec\theta \tag{7.7}$$

$$\theta = q\cos\phi - r\sin\phi \tag{7.8}$$

$$\phi = p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \tag{7.9}$$

作用在机体质心上的力和力矩（力和力矩合称为力素）包括旋翼、尾桨、机身和升力面的气动力素，同时还包括有关旋转部件（如旋翼桨毂系统、尾桨桨毂系统和发动机）的陀螺力矩的贡献。各部件力素可根据当地气流场、速度、加速度分别在当地坐标系下计算或从风洞试验得到，然后通过坐标系变换转换到机体系中。在计算各部件力素的过程中，不仅涉及机体坐标系，还涉及了气流坐标系、桨毂一机体坐标系、桨毂一气流坐标系、桨毂旋转坐标系及桨叶展向坐标系。

这样可以得到作用于机体质心上的力素在机体坐标系 3 个轴上的分量为：

$$X = X_{MR} + X_{TR} + X_F + X_{VS} + X_{HS} \tag{7.10}$$

$$Y = Y_{MR} + Y_{TR} + Y_F + Y_{VS} + Y_{HS} \tag{7.11}$$

$$Z = Z_{MR} + Z_{TR} + Z_F + Z_{VS} + Z_{HS} \tag{7.12}$$

$$L = L_{MR} + L_{TR} + L_F + L_{VS} + L_{HS} \tag{7.13}$$

$$M = M_{MR} + M_{TR} + M_F + M_{VS} + M_{HS} \tag{7.14}$$

$$N = N_{MR} + N_{TR} + N_F + N_{VS} + N_{HS} \tag{7.15}$$

式（7.10）~式（7.15）中，下标“*MR*”、“*TR*”、“*F*”、“*VS*”和“*HS*”分别代表旋翼、尾桨、机身、垂直安定面和水平安定面。

直升机飞行控制时，有总距、纵向、横向和航向 4 个操纵输入，通过自动倾斜器和尾桨桨距机构分别控制旋翼的总距、纵向变距、横向变距以及尾桨的总距，从而操纵直升机的升降、俯仰、滚转和航向。旋翼是直升机的主升力面，产生使直升机飞行最重要的空气动力，但同时带来了操纵耦合的问题，例如，垂直方向的操纵本是改变旋翼的拉力，实现直升机的升降运动，然而拉力的改变会同时造成直升机俯仰力矩的变化，引起了纵向运动。假设在控制向量中，θ_0 为旋翼总距角，θ_{1s} 为旋翼纵向变距角，θ_{1c} 为旋翼横向变距角，θ_{0T} 为尾桨总距角。则可以得到直升机的非线性动力学模型为：

$$x = f(x,u,t) \tag{7.16}$$

其中状态变量为：$x = [u,w,q,\theta,v,p,\phi,r,\psi]^T$，控制变量为 $u = [\theta_0,\theta_{1s},\theta_{1c},\theta_{0T}]^T$。

基于小扰动理论，非线性力可以表示为 $X = X_e + \Delta X$，得到如下形式：

$$X = X_e + \frac{\partial X}{\partial u}\Delta u + \frac{\partial X}{\partial w}\Delta w + \frac{\partial X}{\partial q}\Delta q + \frac{\partial X}{\partial \theta}\Delta\theta + \frac{\partial X}{\partial v}\Delta v + \frac{\partial X}{\partial p}\Delta p + \frac{\partial X}{\partial \phi}\Delta\phi + \frac{\partial X}{\partial r}\Delta r +$$
$$\frac{\partial X}{\partial \psi}\Delta\psi + \frac{\partial X}{\partial \theta_0}\Delta\theta_0 + \frac{\partial X}{\partial \theta_{1s}}\Delta\theta_{1s} + \frac{\partial X}{\partial \theta_{1c}}\Delta\theta_{1c} + \frac{\partial X}{\partial \theta_{0T}}\Delta\theta_{0T}$$

得到

$$X = X_e + X_u\Delta u + X_w\Delta w + X_q\Delta q + X_\theta\Delta\theta + X_v\Delta v + X_p\Delta p + X_\phi\Delta\phi + X_r\Delta r + X_\psi\Delta\psi +$$
$$X_{\theta_0}\Delta\theta_0 + X_{\theta_{1s}}\Delta\theta_{1s} + X_{\theta_{1c}}\Delta\theta_{1c} + X_{\theta_{0T}}\Delta\theta_{0T}$$

其中，X_e、X_u、X_w、X_q、X_θ、X_v、X_p、X_ϕ、X_r、X_ψ、X_{θ_0}、$X_{\theta_{1s}}$、$X_{\theta_{1c}}$、$X_{\theta_{0T}}$ 等为平衡点的气动导数。经过进一步推导，可以得到直升机纵向通道 6doF 的线性化运动方程为：

$$x = Ax + Bu \tag{7.17}$$

其中：

$$A = \begin{bmatrix} X_u & X_w - Q_e & X_q - W_e & -g\cos\Theta_e & X_v + R_e & X_p & 0 & X_r - V_e \\ Z_u - Q_e & Z_w & Z_q + U_e & -g\cos\Phi_e\sin\Theta_e & Z_v - P_e & Z_p - V_e & -g\sin\Phi_e\sin\Theta_e & Z_r \\ M_u & M_w & M_q & 0 & M_v & X_p & 0 & X_r - V_e \\ 0 & 0 & \cos\Phi_e & 0 & 0 & \begin{matrix} M_p + 2P_e I_{xz}/I_{yy} \\ -R_e(I_{xx} - I_{zz})/I_{yy} \end{matrix} & 0 & \begin{matrix} M_r + 2R_e I_{xz}/I_{yy} \\ -P_e(I_{xx} - I_{zz})/I_{yy} \end{matrix} \\ Y_u - R_e & Y_w + P_e & Y_q & -g\sin\Phi_e\sin\Theta_e & Y_v & 0 & -\Omega_a\cos\Theta_e & -\sin\Phi_e \\ L_u' & L_w' & L_q' - k_1 P_e - k_2 R_e & 0 & L_v' & L_p' - k_1 Q_e & 0 & L_r' - k_2 Q_e \\ 0 & 0 & \sin\Phi_e\tan\Theta_e & \Omega_a\sec\Theta_e & 0 & 1 & 0 & \cos\Phi_e\tan\Theta_e \\ N_u' & N_w' & N_q' - k_1 R_e - k_3 P_e & 0 & N_v' & N_p' - k_3 Q_e & 0 & N_r' - k_1 Q_e \end{bmatrix}$$

$$B = \begin{bmatrix} X_{\theta_0} & X_{\theta_{1s}} & X_{\theta_{1c}} & X_{\theta_{0T}} \\ Z_{\theta_0} & Z_{\theta_{1s}} & Z_{\theta_{1c}} & Z_{\theta_{0T}} \\ M_{\theta_0} & M_{\theta_{1s}} & M_{\theta_{1c}} & M_{\theta_{0T}} \\ 0 & 0 & 0 & 0 \\ Y_{\theta_0} & Y_{\theta_{1s}} & Y_{\theta_{1c}} & Y_{\theta_{0T}} \\ L_{\theta_0}' & L_{\theta_{1s}}' & L_{\theta_{1c}}' & L_{\theta_{1T}}' \\ 0 & 0 & 0 & 0 \\ N_{\theta_0}' & N_{\theta_{1s}}' & N_{\theta_{1c}}' & N_{\theta_{1T}}' \end{bmatrix}$$

其中：$X_i = X_i / M_a$，$Y_i = Y_i / M_a$，$Z_i = Z_i / M_a$，

$$L_i' = \frac{I_{zz}}{I_{xx}I_{zz} - I_{xz}^2}L_i + \frac{I_{xz}}{I_{xx}I_{zz} - I_{xz}^2}N_i,\quad N_i' = \frac{I_{xz}}{I_{xx}I_{zz} - I_{xz}^2}L_i + \frac{I_{xx}}{I_{xx}I_{zz} - I_{xz}^2}N_i,\quad \Omega_a = \psi,$$

$$k_1 = \frac{I_{xz}(I_{zz} + I_{xx} - I_{yy})}{I_{xx}I_{zz} - I_{xz}^2},\quad k_2 = \frac{I_{xz}(I_{zz} + I_{xx} - I_{yy})}{I_{xx}I_{zz} - I_{xz}^2},\quad k_3 = \frac{I_{zz}(I_{zz} - I_{yy}) + I_{xz}^2}{I_{xx}I_{zz} - I_{xz}^2}$$

下面以法国航宇工业公司研制的 SA330 Puma 美洲豹直升机为对象，对以上模型进行

仿真验证。SA330 Puma 是双发中型多用途运输直升机，机身背部并列安装 2 台“透默”IV.C 型涡轮轴发动机，最大功率 1600 轴马力左右，法国国内和陆海空三军以及英国皇家空军部队等都大量采购。根据直升机在海平面上（$\rho=1.227$）从盘旋到 140 节的气动导数图（无转弯）可知，直升机在时速为 100 结（52m/s）时的气动导数值为：

$X_u=-0.0275\,1/s$ $\quad Z_u=-0.026\,1/s$ $\quad M_u=0.03\,rad/(s\cdot m)$ $\quad X_{\theta_0}=-2.5\,m/(s^2\cdot rad)$ $\quad L_{\theta_0}=-7.5\,1/s^2$

$X_v=0.006\,1/s$ $\quad Z_v=0.03\,1/s$ $\quad M_v=-0.004\,rad/(s\cdot m)$ $\quad X_{\theta_{1s}}=-9.4\,m/(s^2\cdot rad)$ $\quad L_{\theta_{1s}}=-3\,1/s^2$

$X_w=0.004\,1/s$ $\quad Z_w=0.004\,1/s$ $\quad M_w=-0.025\,rad/(s\cdot m)$ $\quad X_{\theta_{1c}}=0.5\,m/(s^2\cdot rad)$ $\quad L_{\theta_{1c}}=5.3\,1/s^2$

$X_p=0.33\,m/(s\cdot rad)$ $\quad Z_p=0.85\,m/(s\cdot rad)$ $\quad M_p=-0.75\,1/s$ $\quad X_{\theta_{0T}}=0$ $\quad L_{\theta_{0T}}=5.3\,1/s^2$

$X_q=1.5\,m/(s\cdot rad)$ $\quad Z_q=1.5\,m/(s\cdot rad)$ $\quad M_q=-0.75\,1/s$ $\quad Y_{\theta_0}=-3.2\,m/(s^2\cdot rad)$ $\quad M_{\theta_0}=2.5\,1/s^2$

$X_r=0$ $\quad Z_r=0$ $\quad M_r=0$ $\quad Y_{\theta_{1s}}=-0.7\,m/(s^2\cdot rad)$ $\quad M_{\theta_{1s}}=7.5\,1/s^2$

$Y_u=-0.005\,1/s$ $\quad L_u=-0.005\,m/(s\cdot rad)$ $\quad N_u=0.008\,rad/(s\cdot m)$ $\quad Y_{\theta_{1c}}=10\,m/(s^2\cdot rad)$ $\quad M_{\theta_{1c}}=0$

$Y_v=-0.133\,1/s$ $\quad L_v=-0.045\,m/(s\cdot rad)$ $\quad N_v=0.02\,rad/(s\cdot m)$ $\quad Y_{\theta_{0T}}=5\,m/(s^2\cdot rad)$ $\quad M_{\theta_{0T}}=0$

$Y_w=0.0225\,1/s$ $\quad L_w=-0.055\,m/(s\cdot rad)$ $\quad N_w=0.035\,rad/(s\cdot m)$ $\quad Z_{\theta_0}=-115\,m/(s^2\cdot rad)$ $\quad N_{\theta_0}=-5\,1/s^2$

$Y_p=-1.5\,m/(s\cdot rad)$ $\quad L_p=-1.66\,1/s$ $\quad N_p=0$ $\quad Z_{\theta_{1s}}=-41\,m/(s^2\cdot rad)$ $\quad N_{\theta_{1s}}=1.67\,1/s^2$

$Y_q=0.33\,m/(s\cdot rad)$ $\quad L_q=1\,1/s$ $\quad N_q=-0.25\,1/s$ $\quad Z_{\theta_{1c}}=0$ $\quad N_{\theta_{1c}}=0$

$Y_r=-52\,m/(s\cdot rad)$ $\quad L_r=-0.35\,1/s$ $\quad N_r=-0.625\,1/s$ $\quad Z_{\theta_{0T}}=0$ $\quad N_{\theta_{0T}}=-9.5\,1/s^2$

$I_{xx}=9638kg\cdot m^2$ $\quad I_{xz}=2226kg\cdot m^2$ $\quad I_{yy}=33240kg\cdot m^2$ $\quad I_{zz}=25889kg\cdot m^2$ $\quad \Omega_a=27\,rad/s$

我们通过式（7.17）可以得到直升机的特征值、固有阻尼和自然频率如表 7-4 所示。

表 7-4　直升机的特征值、固有阻尼和自然频率

特　征　值	固 有 阻 尼	自然频率（rad/s）
7.15×10^{-4}	-1	7.15×10^{-4}
$2.02\times10^{-4}+27.0j$	-7.50×10^{-6}	27.0
$2.02\times10^{-4}-27.0j$	-7.50×10^{-6}	27.0
$-6.99\times10^{-4}+1.25\times10^{-3}j$	0.488	1.43×10^{-3}
$-6.99\times10^{-4}-1.25\times10^{-3}j$	0.488	1.43×10^{-3}
-6.44×10^{-2}	1	6.44×10^{-2}
-0.122	1	-0.122
-0.693	1	-0.693

可以看出存在多个正的特征根，表明直升机是非线性的不稳定系统。为此，基于该直升机小扰动模型的控制率采用线性二次型最优控制（Linear Quadratic Regulator, LQR），由 Matlab 软件实现，求得状态反馈增益阵为：

$$K=\begin{bmatrix} 0.2937 & -0.9585 & -0.1035 & 0.0220 & -0.0391 & -0.0145 & 0.5150 & 0.0555 \\ -0.9620 & -0.2837 & 0.3646 & 0.7292 & 0.0489 & 0.0392 & -1.6046 & -0.4184 \\ 0.0973 & -0.0227 & 0.0809 & 0.2695 & 0.8805 & 0.3861 & 0.1500 & -1.7452 \\ -0.0630 & -0.0106 & -0.0628 & 0.1518 & 0.4441 & -1.0105 & 0.0407 & -16.8467 \end{bmatrix}$$

并得到基于 LQR 控制后的直升机线性化飞行控制系统动力学模型：

$$\begin{aligned} \dot{x} &= [A - BK]x + Bu \\ z &= Cx \end{aligned} \tag{7.18}$$

其中传感器测量增益阵为 $C = \text{diag}[297.5\ \ 85.0\ \ 85.0\ \ 85.0\ \ 85.0\ \ 85.0\ \ 85.0\ \ 85.0]$。

当控制量 $u = [3\ \text{-}10\ 0.7\ 0]$ 时，得到未经控制的阶跃响应如图 7-20 所示。

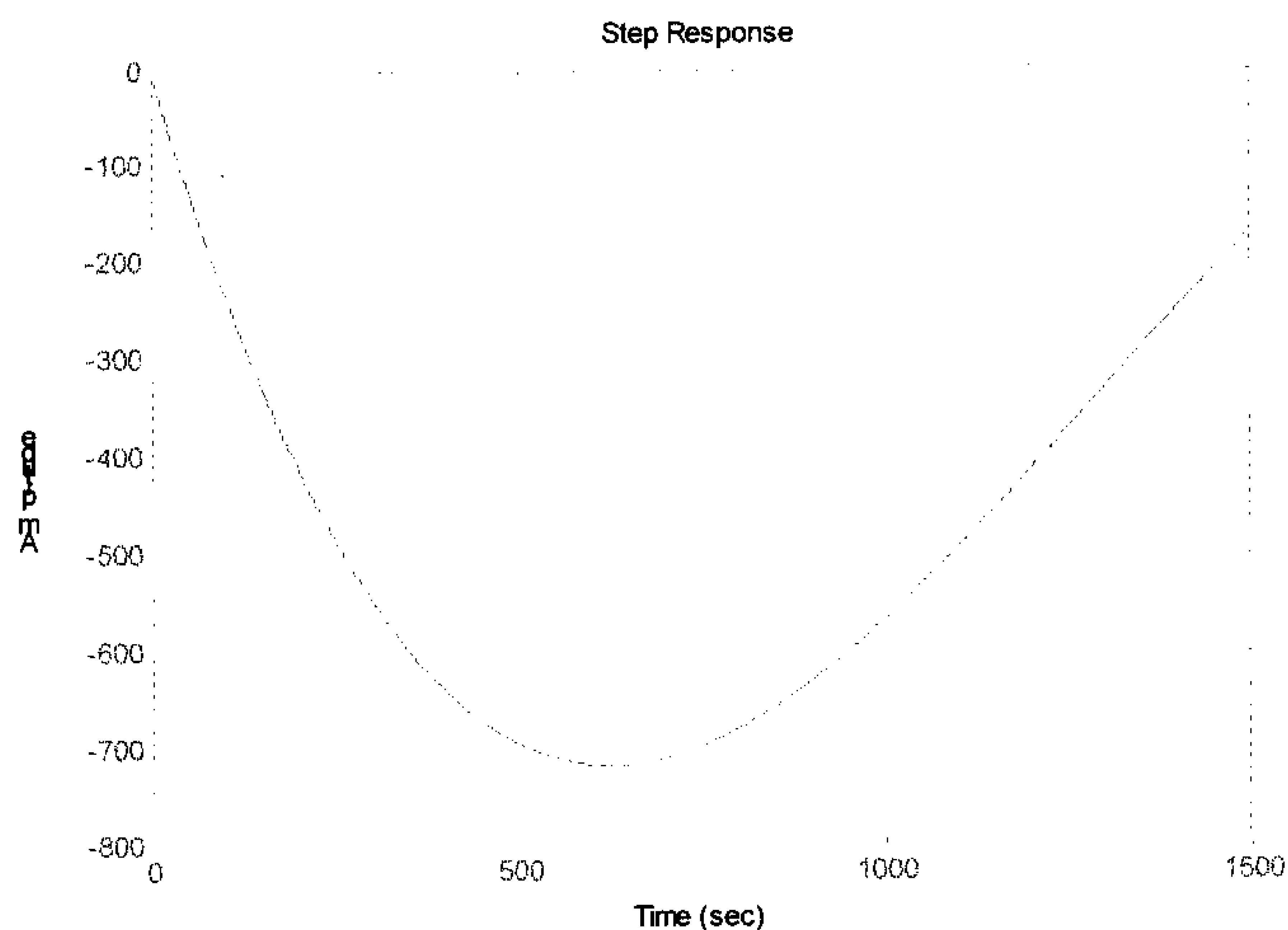


图 7-20　无控制的直升机阶跃响应

经过 LQR 控制的系统阶跃响应如图 7-21 所示。可以看出，经过反馈控制，直升机能够稳定，图 7-22 是直升机各状态传感器的输出变化曲线。

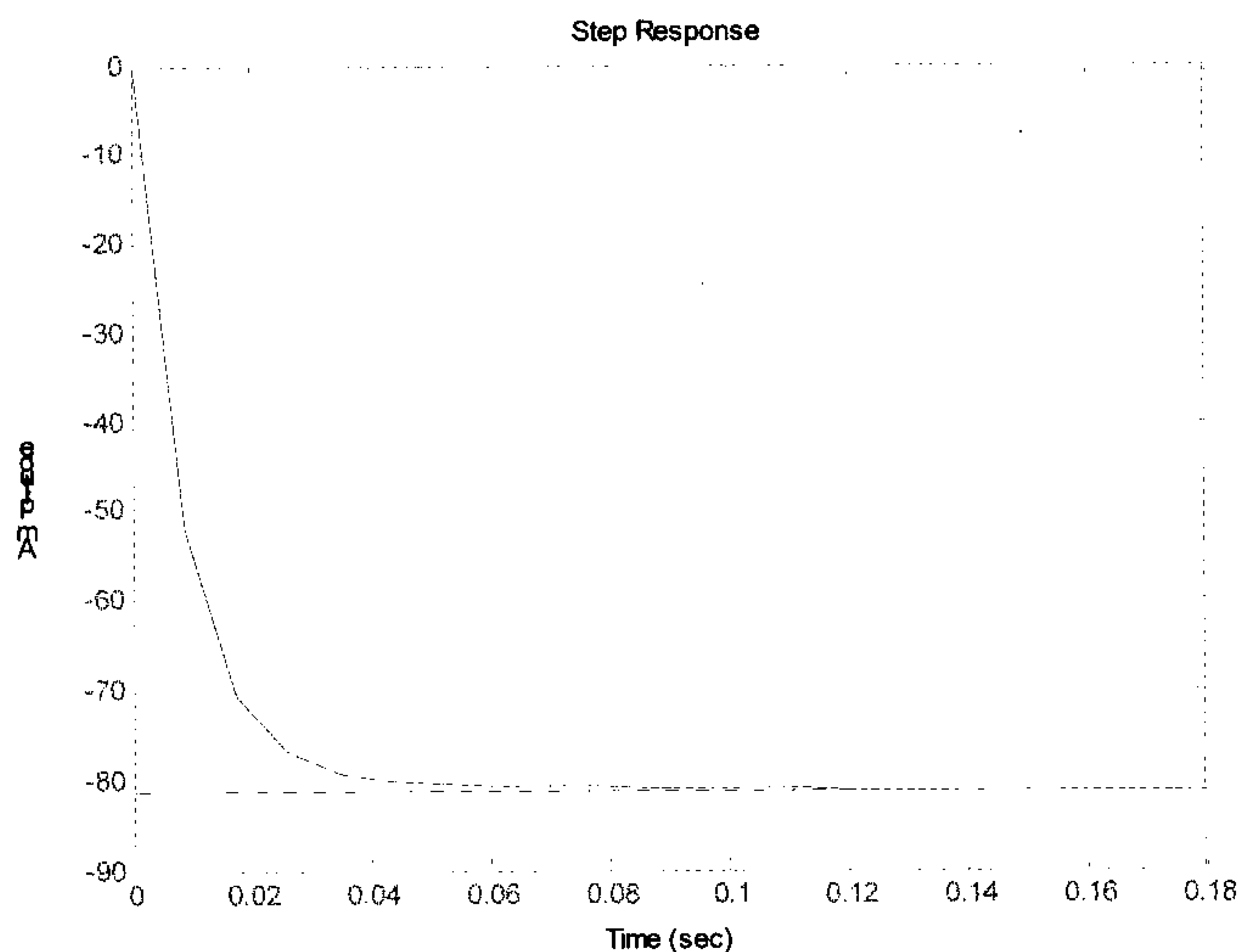


图 7-21　经 LQR 控制的直升机阶跃响应

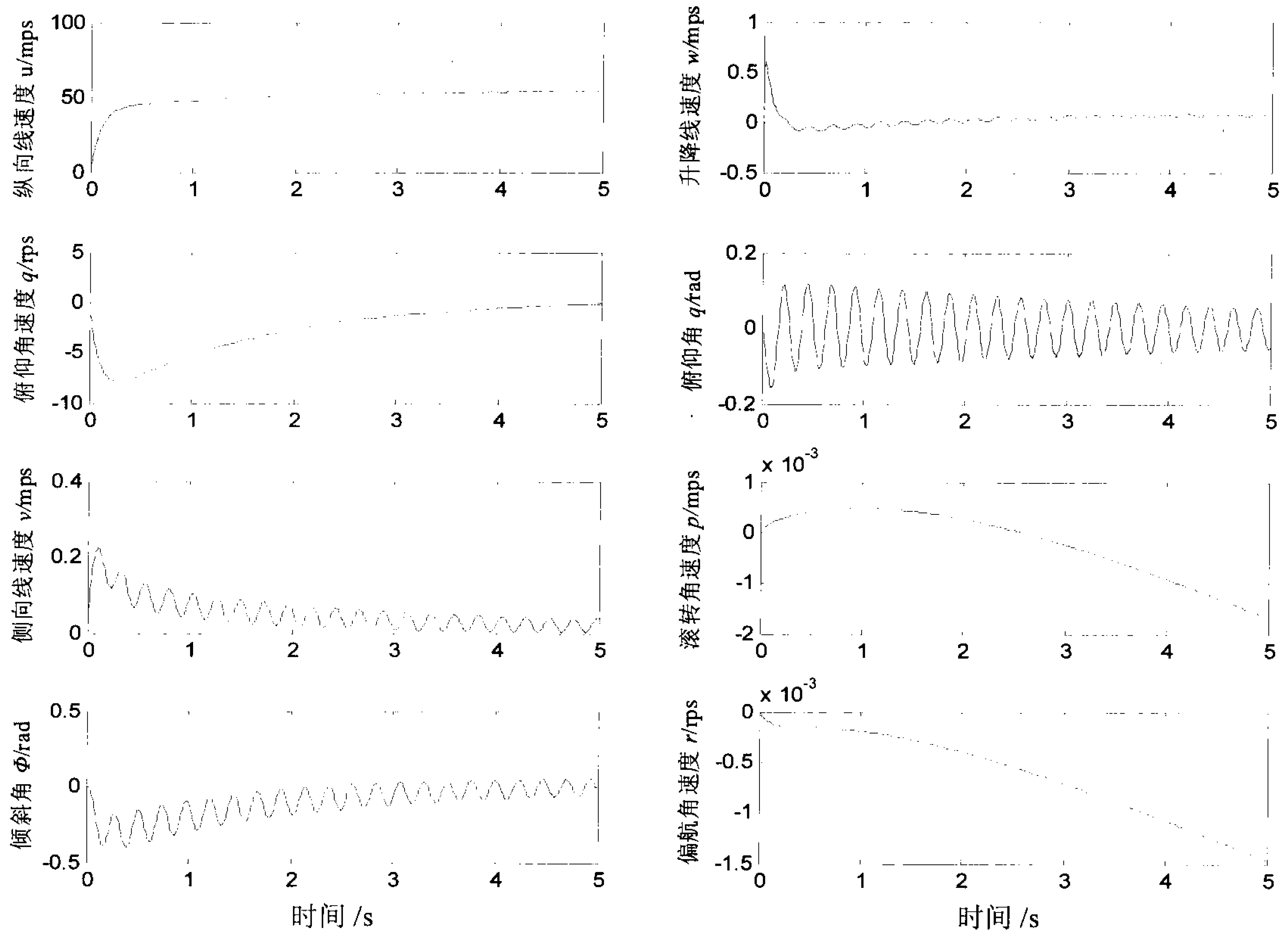


图 7-22　各传感器的变化曲线

整个系统的 MATLAB 实现程序如下：

```
clear;clc;
% 常量
g = 9.81;      % N/s2
Ma = 5805;    % kg
% 惯量
Ixx=9638;     % kg.m2
Iyy=33240;Izz=25889;Ixz=2226;
Ue=52;Ve=0;We=0;Pe=0;Qe=0;Re=0;
THe=0;PHIe=0;
OMEGAa = 27;        %rad/s
% 稳态导数
Xu=-0.0275;Xv=0.006;Xw=0.004;Xp=0.33;Xq=1.5;Xr=0;
Yu=-0.005;Yv=-0.133;Yw=-0.0225;Yp=-1.5;Yq=0.33;Yr=-52;
Zu=-0.026;Zv=0.03;Zw=0.004;Zp=0.85;Zq=1.5;Zr=0;
Lu=-0.005;Lv=-0.045;Lw=-0.055;Lp=-1.66;Lq=1;Lr=0.35;
Mu=0.03;Mv=-0.004;Mw=-0.025;Mp=-0.22;Mq=-0.75;Mr=0;
Nu=0.008;Nv=0.02;Nw=0.035;Np=0;Nq=-0.25;Nr=-0.625;
% 控制导数
Xth0=-2.5;Xth1s=-9.4;Xth1c=0.5;Xth0T=0;
Yth0=-3.2;Yth1s=-0.7;Yth1c=10;Yth0T=5;
Zth0=-115;Zth1s=-41;Zth1c=0;Zth0T=0;
```

```
Lth0=-7.5;Lth1s=-3;Lth1c=25;Lth0T=5.3;
Mth0=2.5;Mth1s=7.5;Mth1c=0;Mth0T=0;
Nth0=-5;Nth1s=1.67;Nth1c=0;Nth0T=-9.5;
% semi-normalize forces
Xu=Xu/Ma;Xv=Xv/Ma;Xw=Xw/Ma;Xp=Xp/Ma;Xq=Xq/Ma;Xr=Xr/Ma;
% Dependent Variables
Lu_pri = (Izz/(Ixx*Izz-Ixz^2))*Lu + (Ixz/(Ixx*Izz-Ixz^2))*Nu;
Lv_pri = (Izz/(Ixx*Izz-Ixz^2))*Lv + (Ixz/(Ixx*Izz-Ixz^2))*Nv;
Lw_pri = (Izz/(Ixx*Izz-Ixz^2))*Lw + (Ixz/(Ixx*Izz-Ixz^2))*Nw;
Lp_pri = (Izz/(Ixx*Izz-Ixz^2))*Lp + (Ixz/(Ixx*Izz-Ixz^2))*Np;
Lq_pri = (Izz/(Ixx*Izz-Ixz^2))*Lq + (Ixz/(Ixx*Izz-Ixz^2))*Nq;
Lr_pri = (Izz/(Ixx*Izz-Ixz^2))*Lr + (Ixz/(Ixx*Izz-Ixz^2))*Nr;

Nu_pri = (Ixz/(Ixx*Izz-Ixz^2))*Lu + (Ixx/(Ixx*Izz-Ixz^2))*Nu;
Nv_pri = (Ixz/(Ixx*Izz-Ixz^2))*Lv + (Ixx/(Ixx*Izz-Ixz^2))*Nv;
Nw_pri = (Ixz/(Ixx*Izz-Ixz^2))*Lw + (Ixx/(Ixx*Izz-Ixz^2))*Nw;
Np_pri = (Ixz/(Ixx*Izz-Ixz^2))*Lp + (Ixx/(Ixx*Izz-Ixz^2))*Np;
Nq_pri = (Ixz/(Ixx*Izz-Ixz^2))*Lq + (Ixx/(Ixx*Izz-Ixz^2))*Nq;
Nr_pri = (Ixz/(Ixx*Izz-Ixz^2))*Lr + (Ixx/(Ixx*Izz-Ixz^2))*Nr;

Lth0_pri = (Izz/(Ixx*Izz-Ixz^2))*Lth0 + (Ixz/(Ixx*Izz-Ixz^2))*Nth0;
Lth1s_pri = (Izz/(Ixx*Izz-Ixz^2))*Lth1s + (Ixz/(Ixx*Izz-Ixz^2))*Nth1s;
Lth1c_pri = (Izz/(Ixx*Izz-Ixz^2))*Lth1c + (Ixz/(Ixx*Izz-Ixz^2))*Nth1c;
Lth0T_pri = (Izz/(Ixx*Izz-Ixz^2))*Lth0T + (Ixz/(Ixx*Izz-Ixz^2))*Nth0T;

Nth0_pri = (Ixz/(Ixx*Izz-Ixz^2))*Lth0 + (Ixx/(Ixx*Izz-Ixz^2))*Nth0;
Nth1s_pri = (Ixz/(Ixx*Izz-Ixz^2))*Lth1s + (Ixx/(Ixx*Izz-Ixz^2))*Nth1s;
Nth1c_pri = (Ixz/(Ixx*Izz-Ixz^2))*Lth1c + (Ixx/(Ixx*Izz-Ixz^2))*Nth1c;
Nth0T_pri = (Ixz/(Ixx*Izz-Ixz^2))*Lth0T + (Ixx/(Ixx*Izz-Ixz^2))*Nth0T;

k1 = Ixz*(Izz+Ixx-Iyy)/(Ixx*Izz-Ixz^2);
k2 = (Izz*(Izz-Iyy)+Ixz^2)/(Ixx*Izz-Ixz^2);
k3 = (Ixx*(Iyy-Ixx)+Ixz^2)/(Ixx*Izz-Ixz^2);

% A & B Matrices
A = [Xu      Xw-Qe      Xq-We                -g*cos(THe)          Xv+Re          Xp
 0                 Xr-Ve;...
    Zu-Qe    Zw          Zq                 -g*cos(PHIe)*sin(THe)    Zv-Pe       Zp-Ve
 -g*sin(PHIe)*sin(THe)    Zr
    Mu       Mw          Mq                        0                           Mv
Mp+2*Pe*Ixz/Iyy-Re*(Ixx-Izz)/Iyy    0              Mr+2*Re*Ixz/Iyy-Pe*(Ixx-Izz)/Iyy
    0        0           cos(THe)            0                  0            0
-OMEGAa*cos(THe)          -sin(THe)
    Yu-Re    Yw+Pe    Yq          -g*sin(PHIe)*sin(THe)     Yv           Yp+We
 g*cos(PHIe)*cos(THe)     Yr-Ue
    Lu_pri   Lw_pri      Lq_pri-k1*Pe-k2*Re        0          Lv_pri        Lp_pri-k1*Qe
 0                  Lr_pri-k2*Qe
    0        0        sin(PHIe)*tan(THe)      OMEGAa*sec(THe)      0              1
 0                cos(PHIe)*tan(THe)
```

```
    Nu_pri   Nw_pri    Nq_pri-k1*Re-k3*Pe    0              Nv_pri        Np_pri-k3*Qe
0                   Nr_pri-k1*Qe];

B = [Xth0        Xth1s        Xth1c        Xth0T;...
        Zth0         Zth1s        Zth1c        Zth0T
        Mth0         Mth1s        Mth1c        Mth0T
        0            0            0            0
        Yth0         Yth1s        Yth1c        Yth0T
        Lth0_pri     Lth1s_pri    Lth1c_pri    Lth0T_pri
        0            0            0            0
        Nth0_pri     Nth1s_pri    Nth1c_pri    Nth0T_pri];

    C = diag([ones(1,8)]);
    D=zeros(8,4);

    [K]= lqr (A,B,C,diag([1 1 1 1]));
    figure(1);
    [n1,d]=ss2tf(A,B,C,D,1);
    [n2,d]=ss2tf(A,B,C,D,2);
     [n3,d]=ss2tf(A,B,C,D,3);
     [n4,d]=ss2tf(A,B,C,D,4);

    tf(n1(1,:),d)
    step(n1(1,:),d)
    p=85;
    figure(2);
     [n1,d]=ss2tf(A-B*K,B,p*C,D,1);
     [n2,d]=ss2tf(A-B*K,B,p*C,D,2);
     [n3,d]=ss2tf(A-B*K,B,p*C,D,3);
     [n4,d]=ss2tf(A-B*K,B,p*C,D,4);
    tf(n1(2,:),d)
    step(n1(2,:),d)
    t=0:0.01:5;
    u=zeros(4,size(t,2));
    for i=1:size(t,2)
        u(:,i)=[3 -10 0.7 0]'*pi/180; % u(:,i)=[5 1.5 0 0]'*pi/180;
    end
    AA=A-B*K;BB=B;CC=p*C;CC(1,1)=297.5;DD=D;
    x0=zeros(1,8);
    sys1=ss(AA,BB,CC,DD);
     [y1,t,x]=lsim(sys1,u,t,x0);   %   基于连续模型;
    % sysd =c2d(sys1,0.001,'foh')
    %[y,t,x]=lsim(sysd,u,t,x0);     %   基于离散模型;
    figure(3);
    for i=1:8
        subplot(4,2,i);plot(t,y1(:,i),'K','linewidth',0.5);
    end
```

第 8 章　MATLAB 高级程序设计技术

通过前面的学习已经知道，MATLAB 语言提供了各种功能强大的函数库，为了调用这些函数实现用户所需的计算和仿真功能，需要进行程序设计、调试和运行。相对于其他语言，MATLAB 具有编程效率高、易学易用等特点，但为了设计功能可读性强、运行效率高、功能完善的应用程序，读者仍然要进行系统性的学习和理解。因此，接下来本章将详细讲述利用 MATLAB 语言进行程序设计的技术和方法，并介绍一些基本技巧。

本章首先循序渐进地介绍 M 文件、MATLAB 的控制流、MATLAB 对面向对象编程的支持；然后结合实例，讲解数据导入导出以及如何改善程序的运行效率和内存使用；结合编程经验，给出了若干编程技巧与提示。调试对于编程是不可或缺的，最后介绍了使用定时器来规划程序的执行。熟悉其他编程语言的用户，如熟悉 Pascal、C++和 FORTRAN 等，会更容易理解本章的内容。

本章主要内容：

- M 文件编程
- 循环、分支、条件语句
- 函数调用与变量传递
- MATLAB 的面向对象编程
- 提高运行效率和改善内存利用
- M 文件的调试与编程技巧

8.1　M 文件编程基础

计算机程序就是计算机指令的集合，不同语言的指令和功能是不一样的。MATLAB 提供了丰富的编程语言，其程序文件有脚本文件和函数文件两种。本章主要介绍 M 文件的编程基础，包括 M 函数类型、MATLAB 控制流、函数调用以及数据导入导出等。

8.1.1　M 文件简介

MATLAB 语言简练而高效，其程序文件为文本文件，后缀为.m，因此 MATLAB 的程序文件通常称为 M 文件。由于 M 文件是文本文件，可以用各种文本编辑工具来对其进行创建和编辑。如 Windows 的记事本程序（Notepad），甚至 DOS 下的文本编辑程序 EDIT 都可以编程编辑 M 文件，只要保存时后缀为.m 即可。另外，MATLAB 提供了专门的 M 文件编辑器，在菜单或工具栏中可以方便地打开 M 文件编辑器。此外，在 MATLAB 的控制窗口直接输入 edit 命令可以打开 M 文件编辑器，如：

```
edit 或者 edit example.m
```

单独的edit命令打开M文件编辑器。若当前目录下有example.m文件，则edit example.m命令打开并编辑，若不存在，则创建一个以example.m为名的M文件并编辑。

可以先使用which命令查看当前路径中是否有某M文件程序，再调用edit命令：

```
which example.m
```

若存在此文件，则返回其路径：

```
D:matlabR\work\example.m
```

若不存在，则查询失败：

```
'example.m' not found.
```

例 8.1 写一个简单的M文件，给出M文件的基本架构。

fact.m

```
function y = fact(n)                          %函数定义行
%Compute a factorial value.                   %H1 行
%FACT(N) returns the factorial of N,           %帮助文档
%usually dnoted by N!

%Put simply, FACT(N) is PROD(1:N).            %注释
y = prod(1: n);                               %函数体
```

下面详细介绍这几部分。

（1）“函数定义行”告诉MATLAB此文件是函数文件，因此只有函数文件才有函数定义行。其他的部分是脚本文件和函数文件共有的基本要素。

上例的定义行为：

```
function y = fact(n)
```

其中，function是关键字，表示定义一个函数，y 是输出参数，n 为输入参数。fact为函数名，函数名必须以字母开头，可以包含任意的字母和下画线，长度限制和变量的长度一样，其他规则也和变量类似。

另外，函数文件的一般命名为函数名加上“.m”后缀，如上述文件命名为fact.m。虽然MATLAB不要求文件名必须和函数名一样，但是使用相同的名字有利于理解和调用，以免混乱。

“函数定义行”还定义了函数的输入参数和输出参数，若输出参数多于一个，则用中括号括起来，而输入参数用小括号括起来，多个参数之间用逗号隔开，如sphere函数的声明：

```
function [x, y, z] = sphere(theta, phi, rho)
```

若无输出参数，则输出参数留空如：

```
function printresults(x)
```

或用空中括号，如：

```
function [] = printresults(x)
```

（2）“H1”表示help的第1行，紧跟函数定义行，当用户在控制窗口输入“lookfor”查找此函数时，便显示H1行的信息，或在控制窗口输入“help filename”时，H1行作为第1行显示。由于H1行提供了此文件的重要信息，所以H1行尽量言简意赅。

（3）“帮助文档”紧跟H1行，也以“%”开头，到第一行不以“%”开头的文本结束。帮助文档是本函数更为详细的描述。在控制窗口输入“help filename”命令，帮助文档会在H1行后显示。

（4）“注释”解释了函数内部工作的一些细节。如算法的解释，控制流的说明和其他一些为增加可读性而作的注释。注释以“%”开头，可以出现在 M 文件的任何位置。

（5）“函数体”（对于函数文件）或代码体（对于脚本文件）包含实际计算和输出参数赋值，是程序功能的实现部分。主要由函数调用、程序控制流、交互性的输入/输出、计算、变量赋值、注释和空白行等。.

8.1.2　M 文件的分类

M 文件分为两种，一种是脚本文件（Scripts File），另外一种是函数文件（Function File）。两种文件的比较如下：

MATLAB 脚本文件特点如下。

- 通常为一连串指令。
- 无输入参数和返回参数。
- 利用的数据和产生的中间结果都保存在 MATLAB 的基本工作空间。

MATLAB 函数文件特点如下。

- 在扩充 MATLAB 函数库中使用。
- 可以接收参数和返回参数。
- 利用的数据和产生的中间结果都保存在函数本身独立的工作空间中。

由上面的比较可以看出，脚本文件的数据由于保存在 MATLAB 基本工作空间中，所以，容易与其他操作产生的变量混淆，比如，某脚本文件中有下面的语句：

```
average  =  sum / lengh ;
```

而 MATLAB 工作空间中已经存在一个 average 变量，这样，调用此脚本文件后，原 average 变量就被覆盖了。后面的利用到 average 的操作会以为以原来的数值进行运算，从而产生难以察觉的错误。所以，利用脚本文件时，除非需要，变量名字力求不和 MATLAB 工作空间的重复。

1．M 脚本文件

MATLAB 指令类似于 DOS 命令，而脚本文件类似于 DOS 系统中的.bat 批处理文件。即脚本文件是一连串 MATLAB 指令，可以将烦琐的计算或操作放在一个 M 文件里面，每当调用这一连串指令时，只需输入 M 文件名即可，从而简化了操作。

脚本与 MATLAB 会话共享基本工作空间。它们主要是操作工作空间中的数据，也可以在工作空间中产生新的数据，继续进行下一步的运算。

例 8.2　编写脚本文件，根据不同的 theta 用三角函数计算 rho 多次，然后根据 theta 和 rho 的值画图。

petals.m

```
% An M-file script to produce          % Comment lines
% "flower petal" plots
theta = -pi:0.01:pi;                   % Computations
rho(1,:) = 2 * sin(5 * theta) .^ 2;
rho(2,:) = cos(10 * theta) .^ 3;
```

```
rho(3,:) = sin(theta) .^ 2;
rho(4,:) = 5 * cos(3.5 * theta) .^ 3;
for k = 1:4
    polar(theta, rho(k,:))          % Graphics output
    pause
end
```

在编辑器上输入以上程序，文件保存为 petals.m，这时 petals.m 就是一个 MATLAB 脚本。在控制窗口的命令行中输入 petals 运行此脚本。运行结果如图 8-1 所示。

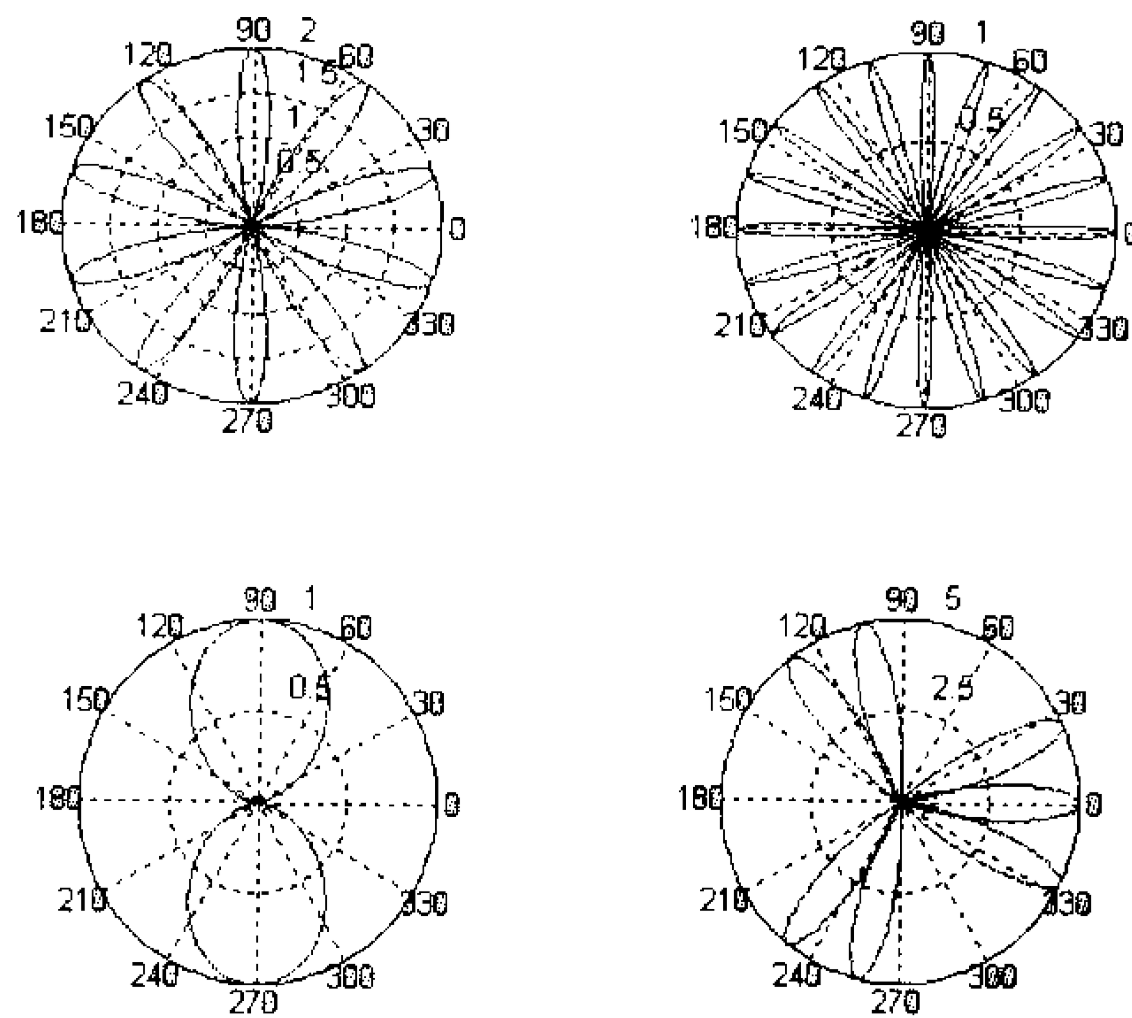


图 8-1 petals.m 的执行结果

由于使用 pause 指令，4 张图片没有同时显示，按回车键，可以让另外 3 张陆续显示出来。此程序没有输入参数和输出参数。但是程序创建了必要的变量，如 theta 和 rho，脚本运行结束后，这些变量会自动保存在 MATLAB 基本工作空间中。

2．M 函数文件

与脚本文件不同，函数文件可以接受输入变量，并可返回结果。M 函数文件通常在扩充 MATLAB 函数库中使用，并且可以接受参数，也可以返回参数，而且不像 C 语言一个函数只能返回一个值，MATLAB 函数可以返回任意多个值。

1）M 函数简介

M 函数文件的实现对于用户来说是透明的。M 函数文件运行时，会创建此函数的函数工作空间（Function Workplace），运算中产生的变量都存放在这个工作空间中，与其他函数空间和 MATLAB 基本工作空间相独立。因此大型的程序宜用函数文件，便于封装与调试。

例 8.3 计算一个向量所有元素的平均值。

average.m

```
function y = average(x)
%  计算向量元素的平均值
%  average(x), 为一个向量 x 元素的平均值,
%  如果没有输入向量，程序将报错

[m,n] = size(x);
if(~((m == 1) | (n == 1)) | (m == 1 & n == 1))
error('please input a vector' )
```

```
end
y    = sum(x) / length(x);                    %  计算
```

这个例子包含了典型的 M 函数的各个部分：函数定义行、H1 行、帮助文档、函数主体以及注释。

函数编辑完成后，将文件保存为 average.m。程序中要求有一个输入参数，在命令行中输入 *z* 个向量并赋值：

```
z = 1:99
```

z =1:99 是定义函数的输入参数，输入文件名调用此函数：

```
average(z)
```

最后输出为：

```
ans  =
50
```

2）M 函数的类型

MATLAB 中的 M 函数有 6 种类型：主函数、子函数、嵌套函数、私有函数、匿名函数以及重载函数。

（1）主函数（Primary M-File Functions）

M 文件中第一个出现的函数称为主函数，其他的函数称为子函数。保存文件时所用的函数文件名与主函数定义名相同。每个 M 文件都要求有一个主函数，它最先出现在该 M 文件中。主函数的范围比子函数的要广，也就是说，主函数可以在 M 文件外部调用（在控制窗口或者其他的 M 文件中），而子函数则不可以。

（2）子函数（Subfunctions）

MATLAB 允许一个 M 文件包含多个函数。同一个 M 文件中除主函数外的其他函数统称为子函数。子函数只在主函数和该 M 文件中的其他子函数中可见。每个子函数都以独自的函数定义行开始。多个子函数可以任意顺序出现，只有主函数必须第一个出现。

例 8.4　计算一个向量所有元素的平均值和中值。

newstats.m

```
function [avg, med] = newstats(u)       %  主函数
% NEWSTATS  利用内部函数计算平均值和中值
n = length(u);
avg = mean(u, n);
med = median(u, n);

function a = mean(v, n)                  %  子函数
%  计算平均值.
a = sum(v)/n;
function m = median(v, n)                %  子函数
%  计算中值
w = sort(v);
if rem(n, 2) == 1
    m = w((n+1) / 2);
else
    m = (w(n/2) + w(n/2+1)) / 2;
end
```

子函数 mean 和 median 分别计算输入参数的平均值和中值。主函数 newstats 计算输入参数的长度 n，传递给子函数，并调用子函数进行计算。

子函数不能访问其他子函数定义的变量，甚至同一个 M 文件的子函数，也不能访问同一个 M 文件的主函数的变量。要访问的话，需要把要访问的变量定义为全局变量或者作为参数传递。

如果想要得到子函数的相关信息，则可以使用 help 命令。例如，如果要获得 M 文件 myfun.m 中的子函数 mysubfun 的信息可以输入：

```
help myfun/mysubfun
```

（3）嵌套函数（Nested Functions）

MATLAB 可以在任意一个函数体内定义一个或多个函数，它们称之为外部函数的嵌套函数。甚至可以在嵌套函数内定义嵌套函数。

嵌套函数也跟其他 M 文件函数一样，包含 M 文件的基本元素。当嵌套函数结束时必须用 end 表示结束。因为成对出现的关键字 function 和 end 使 MATLAB 容易辨别一个完整的函数，不会因为嵌套函数的出现而使函数范围发生混淆。

一个简单的嵌套函数示例如下：

```
function x = A(p1, p2)
...
    function y = B(p3)
    ...
    end
...
end
```

一个函数中，可以嵌套多个函数，如：

```
function x = A(p1, p2)
...
    function y = B(p3)
    ...
    end

    function z = C(p4)
    ...
    end
...
end
```

在一个 M 文件中，还可以使用多重嵌套，如：

```
function x = A(p1, p2)
...
    function y = B(p3)              %B 嵌套在 A 中
    ...
        function z = C(p4)          %C 嵌套在 B 中
        ...
        end
    ...
    end
...
```

```
end
```

结合下面的例子文件 nsetsam.m，嵌套函数有以下调用规则：

- 一个函数可以调用自己的直接嵌套函数。本例中，函数 A 可以调用函数 B 和函数 D，但是不能调用函数 C 或函数 E，因为函数 C 和函数 E 不是函数 A 的直接嵌套函数。
- 嵌套在同一个函数体内的同一级别的嵌套函数可以相互调用。本例中，函数 B 可以调用函数 D，同样，函数 D 可以调用函数 B，但是函数 C 和函数 E 不能互相调用，因为函数 C 和函数 E 不是同一个函数的嵌套函数（直接嵌套）。
- 嵌套函数可以被任意低层的嵌套函数调用。如本例中函数 C 可以调用函数 B 和函数 D，但是不能调用函数 E。

nestsam.m

```
function A(x, y)              %主函数
B(x, y);
D(y);
    function B(x, y)          %嵌套在 A 中
    C(x);
    D(y);
        function C(x)          %嵌套在 B 中
        D(x);
        end
    end
    function D(x)              %嵌套在 A 中
    E(x);
        function E(x)          %嵌套在 D 中
        ...
        end
    end
end
```

每个函数的局部变量往往局限于本函数。例如，子函数不与主函数或其他子函数共享变量，因为每个函数都有独自的工作空间来存储各自的局部变量。嵌套函数虽然也有独自的工作空间，但是它可以访问嵌套它的外部函数的工作空间。例如，外部函数定义的一个变量可以被它的任意层次的嵌套函数读写。同样，一个嵌套函数的变量可以被任何嵌套它的外部函数所读写。

例 8.5 实例说明嵌套函数变量作用域，嵌套函数访问外部函数变量。

varScope1.m

```
function varScope1
x = 5;
nestfun1
    function nestfun1
    nestfun2
    function nestfun2
        x = x + 1
      end
      end
```

```
end
```

其中，变量 x 是外部函数 varScope 的局部变量，但是可由其嵌套函数 nestfun2 所读写。

例 8.6 实例说明嵌套函数变量作用域，外部函数访问嵌套函数变量。

varScope2.m

```
function varScope2
nestfun1
    function nestfun1
    nestfun2
     function nestfun2
     x = 5;
    end
  end
x= x + 1
end
```

其中，x 是嵌套函数 nestfun2 的局部变量，但也可由外部函数 varScope2 所读写。

例 8.7 实例说明嵌套函数变量作用域，外部函数不能访问不确定的嵌套函数变量。

varScope3.m

```
function varScope3
nestfun1
nestfun2
    function nestfun1
        x = 5;
    end
    function nestfun2
        x = x + 1
    end
end
```

其中，外部函数 varScope3 不能访问变量 x，因为嵌套函数 nestfun1 和 nestfun2 是同一级的嵌套函数，其工作空间是独立的，即 nestfun1 的 x 和 nestfun2 的 x 是不同的，当 varScope3 访问 x 时，MATLAB 无法确定要访问哪一个，就会出错。

例 8.8 利用子函数绘制抛物线。

makeParabola.m

```
function fhandle = makeParabola(a, b, c)
% MAKEPARABOLA  返回 parabola 的函数句柄
fhandle = @parabola;
    function y = parabola(x)
    y = a*x.^2 + b*x + c;
    end
end
```

外部函数 makeParabola 创建并返回 parabola 函数的句柄，得到句柄后，利用绘图函数 fplot 进行绘图。

调用 makeParabola 对 parabola 的参数 a,b,c 赋值，并获得嵌套函数句柄：

```
h = makeParabola(1.3, .2, 30);
```

计算抛物线在 x=0 处的函数值：

```
h(0)
ans =
    30
```

最后句柄 h 作为绘图函数 fplot 的参数，进行画图：

```
fplot(h, [-25 25])
```

执行结果如图 8-2 所示。

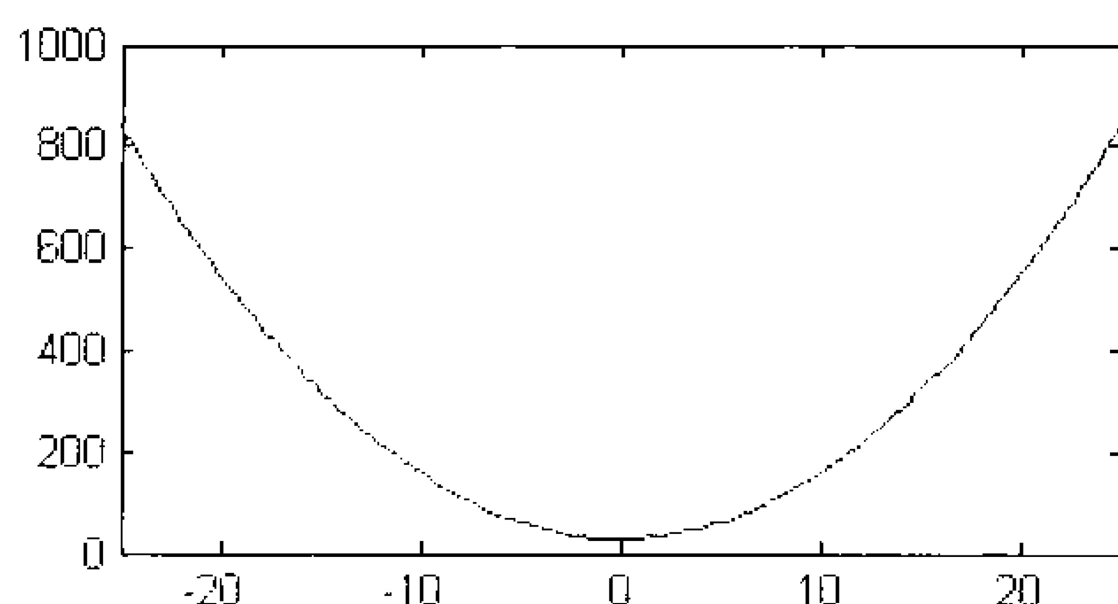


图 8-2　嵌套函数 parabola 所绘抛物线

（4）匿名函数（Anonymous Functions）

匿名函数是 MATLAB 函数的一种简单形式，在 MATLAB 的控制窗口就可以定义和调用，不要求有 M 文件。它只包含一个 MATLAB 表达式、任意多各输入和输出。可以在控制窗口或者 M 函数文件或者脚本文件中定义它。其语法如下：

```
fhandle = @(arglist)expression
```

其中，expression 为此匿名函数的函数体，arglist 为此匿名函数的输入参数列表。等号右边必须以@符号开始，@符号获得此函数的句柄。函数句柄被创建以后，此匿名函数就可通过句柄 fhandle 调用。

下面的表达式创建了一个匿名函数，此函数计算某数的平方值：

```
sqr = @(x)x.^2;
```

例如，可以这样计算 5 的平方值：

```
sqr(5)
ans=
    25
```

因为 sqr 是一个函数句柄，所以，可以将之作为参数传递给别的函数。下面的例子就将 sqr 作为参数传递给了积分函数 quad 进行计算：

```
quad(sqr, 0, 1)
ans
      0.3333
```

匿名函数还可以有多个输入参数，比如，可以在命令行中输入：

```
sumAxBy = @(x, y) (A*x + B*y);
```

A，***B*** 是某已知矩阵，分别给 x，y 赋值 5 和 7，调用此函数：

```
sumAxBy(5, 7)
```

匿名函数也可以不包含任何输入参数，但@后面的参数列表仍必须用空的括号表示，以表示后面表达式是函数，如：

```
t = @()datastr(now);
```

调用此匿名函数同样也要使用括号，如：

```
t()
ans =
     22-Apr-2007 10:23:53
```

否则，MATAB 仅仅识别此句柄，而不会调用此函数，如输入：

```
t
```

则显示：

```
t =
   @()datestr(now)
```

可以建立匿名函数数组，用细胞数组（Cell Array）来存储。如：

```
A = {@(x)x.^2, @(y)y+10, @(x,y)x.^2+y+10}
A =
     [@(x)x.^2]     [@(y)y+10]     [@(x,y)x.^2+y+10]
```

得到细胞数组的前两个函数句柄，可以使用常规的调用数组元素的方法，即 A{1}和 A{2}，得到句柄后，就可以调用函数，如：

```
A{1}(4) + A{2}(7)
ans =
33
```

同样，可以调用第三个匿名函数（两个参数）：

```
A{3}(4, 7)
ans =
     33
```

在函数定义中使用空格可增加函数的可读性，但是在细胞数组中定义的匿名函数体中使用空格却容易产生歧义。

为了使细胞数组中的匿名函数有唯一的解释，可以：

①删除每个函数体的所有空格（参数中的不必要删除）：

```
A = {@(x)x.^2, @(y)y+10, @(x, y)x.^2+y+10};
```

②对有空格的匿名函数用圆括号括起来：

```
A = {(@(x)x .^ 2), (@(y) y +10), (@(x, y) x.^2 + y+10)};
```

③每个匿名函数赋值给一个变量，然后使用这些变量创建细胞数组：

```
A1 = @(x)x .^ 2;   A2 = @(y) y +10;   A3 = @(x, y)x.^2 + y+10;
A = {A1, A2, A3};
```

像 MATLAB 的其他函数一样，匿名函数返回的输出的个数主要决定于调用此函数时等号（=）左边的参数个数。比如，一个匿名函数 getPersInfo 返回一个人的个人信息，按顺序是：地址、家庭电话、工作电话和生日。仅仅获取地址，可以这样调用：

```
address = getPersInfo(name);
```

获取更多的信息，可以用更多的参数：

```
[address, homePhone, busPhone] = getPersInfo(name);
```

（5）私有函数（Private Functions）

私有函数是 M 文件函数的一种。它唯一的特征是只能在一个特定的限定函数群中可见。如果想约束函数的访问，或者选择不让外界看到执行的是哪一个函数的时候，这就很有用了。私有函数放在以 private 命名的子目录下，它们只对其父目录中的函数是可见的。因为私有函数是对外部（父目录外）不可见的，所以，它们可以采用与其他目录下函数相同的名字。MATLAB 先查询私有函数，再查询标准 M 函数。

同样，也可以使用 help 命令来获得私有函数的帮助信息，例如：

```
help private/myprivatefunt
```

（6）重载函数（Reloaded Functions）

重载函数在同一个函数可以接受不同类型或不同个数的参数时非常有用。例如，计算一个区域的面积，可以接受两个 double 型的参数，也可以接受两个 integer 型的参数，还可以接受一个 double 型参数（区域是圆时）。用同一个函数完成所有的面积计算功能，从而实现了 MATLAB 函数的封装，也方便了调用。

具体的重载函数用法，将在面向对象编程中详细介绍。

3）M 函数的句柄

每个函数都有特定的作用域，函数的作用域决定了哪些函数可以访问它，这样有利于函数的封装、维护和安全，但也限制了函数的调用。利用函数句柄可以跨越这个限制，在其作用域外利用函数句柄也可以调用此函数。函数的句柄必须在本函数的作用域内创建，只要能访问到此函数的句柄，就可以利用此句柄来调用函数。

例 8.9　嵌套函数句柄的创建与调用。

getCubeHandle.m

```
function h = getCubeHandle
h = @findCube;                      % Function handle constructor

    function cube = findCube(X)    % Nested function
        cube = X .^ 3;
    end
end
```

函数 getCubeHandle 创建了函数 findCube 的句柄，将之返回给变量输出变量 *h*。符号@是 MATLAB 的操作符，用在函数前表示取此函数句柄值。

调用 getCubeHandle 获取 findCube 函数的句柄，并把此句柄返回给输出变量 cublt。

```
cubeIt = getCubeHandle;
```

此时，在 M 文件的外部可以调用此函数：

```
cubeIt(8)
ans =
     512
```

例 8.10　用句柄来调用子函数和嵌套函数。

sCountFun.m

```
function h = sCountFun(X)
h = @subCount;
count = X
subCount(0, count);
function subCount(incr, ini)
    persistent count;
    initializing = nargin > 1;
    if initializing
      count = ini;
    else count = count + incr
```

```
end
```

nCountFun.m

```
function h = nCountFun(X)
h = @nestCount;
count = X
  function nestCount(incr)
    count = count + incr
  end
end
```

本例中两个函数 sCountFun 和 nCountFun 返回了子函数和嵌套函数的句柄。这两个内部函数都存储了一个永久变量 count，在每次调用内部函数时，使 count 值增加。子函数 subCount 用 persistent 关键字显式地声明 count 为永久变量，在嵌套函数 nestCount 中，变量 count 是外部变量但也同时保存在嵌套函数的句柄中。

当 sCountFun 和 nCountFun 执行，分别初始化了子函数 subCount 和 nestCount 的 count 值。但是注意 sCountFun 和 subCount 中的 count 是完全独立的两个变量，而 nCountFun 和 nestCount 的 count 变量却是同一个。每当 subCount 用句柄调用，会从内存中读取永久变量 count 的值，而 nestCount 用句柄调用时，count 从外部函数的工作空间中读取。

虽然两个 count 变量存储位置不完全相同，但是由于都存储在内存中，所以两调用 subCount 和 nestCount 会得到相同的结果。

如对于子函数例子，初始化 count 为一个四元组：

```
h = sCountFun([100 200 300 400])
count =
       100    200    300    400
```

得到子函数的句柄后，用句柄调用：

```
h(25)
count =
     125    225    325    425
```

对于嵌套函数例子：

```
h = nCountFun([100 200 300 400])
count =
       100    200    300    400
h(25)
count =
       125    225    325    425
```

由于存储在内存中，若再调用，会在结果上继续增值：

```
h(25)
count =
       150    250    350    450
```

8.1.3 MATLAB 控制流

和其他高级语言一样，MATLAB 也提供了对程序控制的支持，从而使得 MATLAB 语言的编程显得十分灵活。MATLAB 共有 4 种控制结构，和 C 语言的控制语句很相似。

- 条件控制结构（Conditional Control）：if、switch
- 循环控制结构（Loop Control）：for、while、continue、break
- 程序结束控制（Termination Control）：return
- 错误控制结构（Error Control）：try-catch

1．条件控制结构

条件语句分为 if 语句和 switch 语句两种。

1）if、else、else if 语句

if 用来检查逻辑运算、逻辑函数、逻辑变量值等逻辑表达式的真假，若为真，则执行接下来的指令或运算。其基本语法如下：

```
if logical_expression
    statements
end
```

例如，如下的程序给 a 赋值。

```
for i = 1:6
   a(i) = i;
   if i > 3
    a(i) = 6 - i ;
   end
end
```

执行结果如下：

```
a =
     1   2   3   2   1   0
```

如下的程序判断 a 是否是偶数，是，则输出判断结果并除以 2。

```
if rem(a, 2) == 0
    disp('a is even')
    b = a/2;
end
```

如果逻辑表达式不是一个标量，那么表达式值为真，当且仅当其所有的元素都为真。例如，假设 ***X*** 是一个矩阵，那么表达式

```
if X
    statements
end
```

等价于

```
if all(X(:))
    statements
end
```

最常见的条件语句是 if-then-else，其语法如下：

```
if 逻辑表达式
   执行语句 true ;
else
   执行语句 false ;
end
```

当逻辑表达式为真（非 0 值）时，则执行语句 true，else 后无逻辑表达式，语句在逻辑表达式非真值（0）时被执行。

MATLAB 还可以执行多项条件，其语法如下：

```
if 逻辑表达式 1
    执行语句 1；
elseif 逻辑表达式 2
    执行语句 2；
elseif 逻辑表达式 3
    执行语句 3
……..
else
    执行语句 n
end
```

其中，一个 if 后可以跟任意多个 else-if 语句。

例 8.11 利用多重 elseif 分支，判断一个数是否能被 2、3、4 整除。

condition.m

```
function condition(n)
if n < 0                          %若 n 是负数，显示错误信息.
    disp('Input must be positive');
elseif rem(n,2) == 0              %若 n 是正偶数，除以 2
    A = n/2;
elseif rem(n,3) == 0
    A = n/3;                      %若 n 能被 3 整除，除以 3
elseif rem(n,4) == 0
    A = n/4;                      %若 n 能被整除，除以 4
else
    disp('A 不能整除 12');
end
```

2）switch 、case、otherwise 语句

如果在一个程序中必须针对某个变量的不同取值来做多种不同的执行，通常使用 switch 语句，其语法如下：

```
switch expression(scalar or string)          %switch 语句的开始，紧接着分支条件
   case value1
      statements                             %当表达式的值等于 value1 时执行该语句
case value2
      statements                             %当表达式的值等于 value2 时执行该语句
……
otherwise
      statements                             %当不符合所有的 case 条件时，则执行 otherwise
                                             %下面的这条语句
end;
```

其中，otherwise 是可选项，case 语句可以有任意多个，后面的执行语句可以是 MATLAB 的任意表达式，也可以是一个 switch 语句，即 switch 可以嵌套，如：

```
switch expression(scalar or string)
   case value1
      switch expression(scalar or string)        % 嵌套 switch
         case valueA
            statements
      end
```

```
......
otherwise
    statements
end;
```

例 8.12　检查变量 input_mun 的值。如果 input_num 等于一1、0 或者 1，那么以文本的形式打印 input_num 的值。如果 input_num 不等于这 3 个值中的任何一个，则程序跳入 otherwise 中打印“other value”。

case_exa.m

```
function case_exa(input_num)
switch input_num
    case -1
        disp('negative one');
    case 0
        disp('zero');
    case 1
        disp('positive one');
    otherwise
         disp('other value');
end
```

switch 语句还可以在一个 case 语句中处理多个条件值，例如：

```
switch var
case 1
  disp('1')
case {2,3,4,}
  disp('2 or 3 or 4')
case 5
  disp('5')
otherwise
  disp('other values')
end;
```

MATLAB 中的 switch 结构与 C 语言的 switch 语句很相似，但也略有差别。在 C 语言中，检验某个 case 符合并执行其运算后，还会再继续检验下一个 case，直到全部检验完。所以，一般会在一个描述的最后加入 break，让程序只运算第一个检验成功的运算式。但是，MATLAB 的 switch 语句只执行第一个检验成功的 case，然后跳出 switch 语句。

2. 循环结构

循环结构有 4 个语句：for、while、continue 和 break。循环结构可以方便地按照任意规定次数重复地执行某个程序块。for 语句一般适用于循环次数已知，或者可简单计算得出的情况；while 语句更适用于循环条件为复杂逻辑表达式，而循环次数预先不易得知的情况；continue 和 break 语句用于使程序跳出循环。

1）for 语句

for 语句的语法如下：

```
for index = 初始值:增量（或步值）:结束值
    statements
end
```

增量值默认为 1，也可以自定义增量值，增量值可以小于 0。当增量值大于 0 时，程序将在变量大于终止值时终止运行。当增量值小于 0 时，程序将在变量小于终止值时结束运行。

例如，下面的 for 循环将执行 5（6－2+1＝5）次。

```
for n = 2 : 6
    x(n) = 2 * x(n-1);
end
```

也可以定义多重嵌套的 for 循环语句。

例 8.13 多重嵌套 for 循环给矩阵各元素赋值。

mul_for.m

```
for m = 1:5
    for n = 1:5
        A(m, n) = 1/(m + n - 1);
    end
end
A
```

循环执行结束后，矩阵 $\boldsymbol{A}$ 的值为：

```
A =
    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111
```

另外，for 语句中的变量也可以是任意合法的 MATLAB 数组。对于一个 $m\times n$ 的矩阵 $\boldsymbol{A}$ 来说，有：

```
for k = A
        运算指令
end
```

该指令中 k 被设置为一维数组 $A(;,\mathrm{i})$，第一次循环，k 被设置为一维数组 $\boldsymbol{A}(;,1)$，第二次循环，k 被设置为 $\boldsymbol{A}(;,2)$，直到 k 被设置为 $A(;,n)$。即 $\boldsymbol{A}$ 每次循环执行时，k 为矩阵 $\boldsymbol{A}$ 中 1 列的所有元素。

值得注意的是，循环语句在 MATLAB 中执行效率很低。由于 for 循环语句的循环次数往往是已知或者易知的，所以为了提高 MATLAB 的执行效率，往往把 for 语句展开。这一点会在后面的“提高编程效率”一节中详细介绍。

2）while 语句

另外一个常用的循环语句是 while 循环，其语法如下：

```
while expression
    statements
end
```

其执行方式为：若表达式为真（非 0 值），则执行循环体的内容。执行完毕后再判断表达式是否为真，若为真，则继续执行循环体的内容，再判断表达式的值，直到表达式值为假才跳出循环体，向下继续执行。

例 8.14　找出第一个阶乘（N!）值是 100 位数的整数。

while_exa.m

```
n = 1;
while prod(1:n) < 1e100
    n = n + 1;
end
n
prod(1:n)
```

执行结果如下：

```
n =
     70
ans =                    % 70 的阶乘（70!）
      1.1979e+100
```

例 8.15　求从 1 到多少个自然数的和大于或等于 100。

findfirst.m

```
sum = 0; i = 0;
while sum<100
    i = i + 1 ;
sum = sum + i ;
end
sum
i
执行结果如下：
sum =
      105
i =
      14
```

3）continue 语句

continue 在 for 或者 while 循环中用于直接跳至下一个循环的执行。在嵌套式循环中，continue 控制的是与自己最近的一个 for 或者 while 循环。

例 8.16　计算文件 magic.m 的行数，空白行和注释行除外。continue 语句相当于在遇到空白行或者注释行时，继续读入 magic.m 中的下一行而不增加行数值。

count_line.m

```
fid = fopen('magic.m', 'r');
count = 0;
while ~feof(fid)
      line = fgetl(fid);
      if isempty(line) | strncmp(line, '%', 1)
            continue
      end
      count = count + 1;
end
disp(sprintf('%d lines', count));
```

4）break 语句

break 语句用于终止 for 循环或者 while 循环。当遇到 break 语句时，程序执行循环体外的下一条语句。在嵌套式循环中，break 语句只是跳出最近的循环语句。下面的 while 循环实例是将文件 fft.m 的内容读入到字符序列中，break 语句用于遇到空白行时退出 while 循环。程序返回的字符序列包括程序 fft.m 的帮助文档。

fft.m

```
fid = fopen('fft.m', 'r');
s = '';
while ~feof(fid)
    line = fgetl(fid);
    if isempty(line)
        break
    end
    s = strvcat(s, line);
end
disp(s)
```

3．程序结束控制

程序结束控制用 return 语句，用于结束 return 指令所在函数的执行，返回到主调函数或者指令窗口。

例 8.17 计算矩阵的特征值，当遇到空的矩阵时，必须使用 reutrn 指令来处理。

det.m

```
function d = det(A)
% det det(A) is the determinant of A.
if isempty(A)
    d = 1
    return
else
.....
end
```

return 也用来结束 keyboard 模式。

4．错误控制结构

错误控制结构使用 try-catch 语句，用来捕捉并处理异常。try 语句用来检测程序代码是否会产生错误，一旦错误发生，MATLAB 会立即跳入到相应的 catch 语句中去，对错误做相应的处理。其语法如下：

```
try
    statement1          %命令块 statement1 被执行。若正确，则跳出此结构
catch
    statement2          %当命令块 statement1 出现执行错误时则执行命令块 statement2
end;
```

例 8.18 用 try-catch 检测两个矩阵的乘法可能出现的错误。

matrixMultiply.m

```
function matrixMultiply(A, B)
try
    X = A * B
catch
    disp '** Error multiplying A * B'
end
```

输入矩阵 ***A***，***B*** 的值，执行上面的程序，当 ***A*** 的列数不等于 ***B*** 的行数时，MATLAB 就会发现这个错误，打印消息“**error multiplying A*B”。

给 ***A***、***B*** 赋值：

```
A = [1   2   3; 6   7   2; 0   1   5];
B = [9   5   6; 0   4   9];
```

调用矩阵乘法函数：

```
matrixMultiply(A, B)
** Error multiplying A * B
```

同样，try-catch 也可以以嵌套方式使用，语法如下：

```
try
  statement1                        %执行命令块 statement1
catch
    try
        statement2                  %恢复错误
    catch
        disp ‘Operation failed’      %处理错误
    end
end;
```

5．控制程序的其他指令

本节将系统地介绍控制指令流中的 input、keyboard、pause、error 和 echo 等指令。

1）input 指令

指令 input 提示用户从键盘输入数值和字符表达式，并接收该输入，语法如下：

- user_entry = input(‘message’)
- user_entry = input(‘message’,’s’)

指令运行后，将给出文字提示 message，并等待键盘输入。

用户可以输入数字或表达式，也可以输入字符串（两端必须有单引号），按【Enter】键确认后，该输入被赋值给变量 user_entry。

两种格式的区别是，不管在第二种格式中输入什么内容，总以字符串的形式赋给变量 user_entry。如果用户输入了 return 指令，而没有输入任何值，则输入返回的是一个空的矩阵。当输入为一个字符串时，可以在输入中使用转义符，例如，/n 代表换行。如果要输入“/”，则要输入“//”。

例 8.19　用 input 获取用户信息。

try_input.m

```
reply = input('Do you want more? Y/N [Y]: ', 's');
if isempty(reply)
```

```
        reply = 'Y';
    end
```

当输入 return 指令时，reply 的值将等于“Y”。

2）keyboard 指令

keyboard 指令与 input 指令同样有用。在程序遇到 keyboard 指令时，MATLAB 将会暂停程序的执行，并且调用机器的键盘命令进行处理。一旦处理完自己的工作之后，输入 return 指令，然后按下【Enter】键，程序将继续执行。它与 input 的区别在于：它允许输入任意多个 MATLAB 指令，而 input 只能给变量赋值。

3）pause 指令

pause 指令使程序运行暂停，语法如下：

- pause：暂停执行程序，等待用户按任意键继续。
- pause(n)：使程序暂停 *m* 秒以后继续执行。*n* 的取值为一非负实数。
- pause on：指令允许连续的 pause 指令暂停程序的执行。
- pause off：指令使连续的 pause 或者 pause(n)指令变得无效，从而使得一些脚本可以自动运行。

pause 语句在绘图程序中非常有用，由于程序执行非常快，若无 pause 中断程序的执行，则绘图的中间过程几乎看不到。如下例所示：

```
x = -pi:pi/20:pi;
plot(x,cos(x))
pause
title('A Short Title')
grid on
```

4）error 指令

error 指令用于显示出错信息，终止程序。error 有以下 5 种语法结构：

- error('message')：显示出错的信息，包含输入的字符消息 message。
- error('message',a1,a2,...)：使用方法与 MATLAB 的 sprintf 相似，显示的消息包含转换格式的字符。
- error('message_id','message')：给错误信息绑定一个唯一的标识符或者消息 ID。标识符能够更好地标识错误源。
- error('message_id','message',a1,a2,...)：在消息中包含了格式转换字符及对字符 a1、a2 的解释。
- error(message _struct)

例 8.20　error 语句用来检查函数的参数。

foo.m

```
function foo(x,y)
if nargin ~= 2
        error('Wrong number of input arguments')
else
        ....
end
```

foo 函数首先检查输入参数的个数，若不等于 2，则显示错误信息，并退出程序执行。

例如，输入一个参数执行 foo：

```
foo(pi)
??? Error using ==> foo at 3
Wrong number of input arguments
```

利用 error 来确定错误信息的标识及对应的错误信息：

```
error('MyToolbox:angleTooLarge', ...
        'The angle specified must be less than 90 degrees.');
```

在错误处理中，用 lasterror 来确定错误信息的标识和对应信息：

```
err = lasterror;
```

查询错误信息：

```
err.message
```

则输出结果如下：

```
ans =
   The angle specified must be less than 90 degrees.
```

查询错误信息标识：

```
err.identifier
```

则输出结果如下：

```
ans =
   MyToolbox:angleTooLarge
```

如果此错误是某 M 文件抛出的，利用 lasterror 的栈还可以查找出此 M 文件、函数以及出错的行数等信息。如：

```
err.stack
ans =
     file: 'd:\mytools\plotshape.m'
     name: 'check_angles'
     line: 26
```

若错误信息中包含 MATLAB 的转义符，比如“\n”和“%d”，则当 error 的参数多于一个时，才能被正确识别。如下例中，“\n”代表换行，但是由于参数只有一个，故没有被识别：

```
error('In this case, the newline \n is not converted.')
??? In this case, the newline \n is not converted.
```

当参数多于一个时，不管其他参数是什么，MATLAB 就能识别了：

```
error('ErrorTests:convertTest', ...
        'In this case, the newline \n is converted.')
??? In this case, the newline
 is converted.
```

5）echo 指令

通常 M 文件执行时，文件的指令不会显示在指令窗口中。用 echo 指令可以使文件指令在执行时可见，这对程序的调试和演示极为有用。对应于脚本文件和函数文件，echo 的作用稍有不同，语法如下：

- echo on：当 echo 状态为 off 时，显示其后所有被执行命令文件的指令，并打开 echo 状态为 on。
- echo off：当 echo 状态为 on 时，显示此语句前的被执行语句，而不显示其后所有被执行命令文件的指令。并使得 echo 状态为 off。
- echo：在上面两种状态之间切换。如上面使用了 echo on 指令，则再用 echo 指令

后，此时相当于使用 echo off 指令，反之亦然。

- echo fcnname on：使 fcnname 指定文件的指令在执行中被显示出来。
- echo fcnname off：终止显示 fcnname 文件的执行过程。
- echo fcnname：在上面第四和第五种状态之间切换。
- echo on all：显示其后所有文件的执行过程。
- echo off all：不显示其后所有文件的执行过程。

前 3 种仅限于脚本文件，而后面的所有指令二者都适用。但是，前 3 种即使用在函数文件中，也不会报错，即函数文件也可以正确执行，只是 echo 语句不被执行而已。

例 8.21 echo on 作用示例。

displayon.m

```
a = 1+2        %不会在控制窗口显示出来
echo on;
c = a+6        %会在控制窗口显示出来
```

若此时 echo 状态为 off，则此脚本执行结果如下：

```
echo on
c = a+6        %会在控制窗口显示出来
```

此时 echo 的状态置为 on，再次执行此脚本时，结果如下：

```
a = 1+2        %不会在控制窗口显示出来
echo on;
c = a+6        %会在控制窗口显示出来
```

例 8.22 echo off 作用示例，在上例执行结束后执行脚本 displyoff.m。

displayoff.m

```
a = 1+2;
echo off
c = a+6;
```

则结果如下：

```
a = 1+2;
```

这是由于在 echo off 语句执行前，echo 的状态为 on，所以，此脚本前面的执行语句照常显示。

8.1.4 函数调用和变量传递

MATLAB 中函数的调用方法与 C 语言中的调用方法相似，可以在控制窗口以命令行形式调用，也可以在 M 函数中调用。而 MATLAB 的参数传递一般是值传递，另外，MATLAB 支持多个返回参数。

1．函数调用

当从命令行或 M 文件调用另外一个 M 文件时，MATLAB 把函数解析成伪码（Pseudocode）并存储在内存中。这样在下次调用此函数时，就不必再次对此函数解析了。除非用 clear 函数清除或退出 MATLAB，否则，此伪码一直会驻留在内存。

可以使用下面几种方法清除内存中的伪码。

- clear functionname：清除指定文件名的伪码。
- clear functions：清除所有 M 函数的伪码。
- clear all：清除内存中所有变量和函数。

1）函数调用顺序

当多个函数有相同的函数名时，MATLAB 将根据函数的位置及类型按顺序调用。MATLAB 调用函数的顺序遵循以下原则。

（1）变量。在进行函数名匹配之前，MATLAB 先在当前工作空间查找是否存在以此为名的变量。若存在，则以为是变量同时结束查找。

（2）子函数。子函数比相同路径上其他的 M 函数和重载函数的优先级要高。

（3）私有函数。

（4）类的构造函数。

（5）重载函数。

（6）当前路径上的函数。

（7）搜索其他路径。

第三条到第七条中的函数都可以是下面 5 种类型之一：M 函数、内嵌函数、P 伪码、MEX 文件和 Simulink model（MDL 文件）。对于这 5 种类型的函数，也有其如下调用顺序。

（1）MATLAB 的内嵌函数。

（2）MEX 文件。

（3）MDL 文件。

（4）P 伪码文件。

（5）用户的 M 文件。

用 which 命令可以查询 MATLAB 会调用哪个函数。比如：

```
which pie3
matlabroot/toolbox/matlab/specgraph/pie3.m
```

2）函数调用语法

命令行方式调用函数的语法如下，函数名后跟参数列表，函数名和参数以及各参数之间用空格符隔开：

```
functionname in1 in2 ... inN
```

命令行的函数语法比较简单易写，但是，缺点是不能为函数的返回参数赋值。

命令行调用函数的简单例子如下：

```
save mydata.mat x y z
clear length width depth
```

函数式的语法和其他编程语言相似，MATLAB 的特别之处在于一个函数可以返回多于一个的参数。

只有一个返回值的函数语法如下：

```
out = functionname(in1, in2, ..., inN)
```

若函数返回多于一个参数，参数之间用逗号或空格隔开，然后用方括号（[]）把所有参数括起来：

```
[out1, out2, ..., outN] = functionname(in1, in2, ..., inN)
```

两个简单例子如下：

```
copyfile(srcfile, '..\mytests', 'writable')
[x1, x2, x3, x4] = deal(A{:})
```

用函数式调用函数时，对参数表中的参数进行赋值，例如，下面的表达式为 polyeig 函数的参数 A0、A1 和 A2 赋值：

```
e = polyeig(A0, A1, A2)
```

而用命令行式调用函数时，参数以文字字符的形式传递，下面的表达式为 save 函数传递了字符串形式的参数'mydata.mat'、'x'、'y'和'z'：

```
save mydata.mat x y z
```

例 8.23 以例子来说明两种传递参数方式的不同。

arguement.m

```
% 先函数式调用 disp 函数：
A = pi;
disp(A)                        % Function syntax
   %显示结果为 pi 的值：3.1416
% 再以命令行形式调用 disp A，传递字符串参数'A'：
A = pi;
disp A                         % Command syntax
% 显示结果为字符'A'：  A
```

以命令行执行 disp A 时，会把 A 当做字符串，而不是变量，所以会显示'A'，而非 A 所代表的内容。

例 8.24 用命令行形式和函数式分别为 strcmp 函数传递两个有相同内容的字符串，查看结果。

strcmpsam.m

```
str1 = 'one';          str2 = 'one';
strcmp(str1, str2)                        % Function syntax
ans =
      1          (相等)
```

用函数式调用 disp 函数时，得到的结果为 1，表示两字符串相等（相同）。

```
str1 = 'one';          str2 = 'one';
strcmp str1 str2                          % Command syntax
ans =
      0          (不相等)
```

用命令行方式调用 disp 函数时，得到的结果为 0，即不相等（相同），因为在这种情况下，由于命令行式调用函数，参数是以字符串形式传递的，所以，strcmp 函数比较的是字符串'str1'和'str2'，而非 str1 和 str2 的值'one'。

2. 参数传递

对函数进行调用时，返回参数个数可以少于函数定义时的返回参数个数，但是不可以多于。比如，一个函数定义有 N 个返回参数，但是调用时，可以使用 0 到 N 个返回参数。不需要的返回参数被丢弃。函数调用时，按照函数定义行指定的顺序来返回参数。

例 8.25 举例说明参数返回的顺序。

myfun.m

```
function [a b c] = myfun(x, y)
b = x * y;
a = 100;
c = x.^2;
```

上例先返回 100，然后是 $x*y$，最后是 $x.\hat{}2$，虽然先计算得到了 $x*y$，但是参数列表中 a 在 b 前面，故先返回 a 的值。

当返回参数个数少于函数定义的返回参数个数时，尤其要注意上面的顺序。如调用时一个返回参数：

```
a = myfun(x, y)
```

此时仅仅返回 100 而不是 $x*y$ 的值，b 和 c 的值被丢弃。如调用时无任何返回参数：

```
myfun(x, y)
ans =
     100
```

1）检查参数个数

与 C 语言一样，MATLAB 可获取输入参数个数信息并根据不同输入参数个数完成不同的功能。MATLAB 中用 nargin 和 nargout 函数获取函数调用时输入参数和输出参数的个数。

例 8.26 对输入参数判断，然后完成不同的任务。

testarg.m

```
function c = testarg(a, b)
if (nargin == 1)
      c = a .^ 2;
elseif (nargin == 2)
      c = a + b;
end
```

2）可选个输入输出参数

利用 varargin 和 varargout 函数可以传递任意数目的输入参数和输出参数。MATLAB 将所有的输入参数打包成一个细胞数组，而输出参数需要自己编写代码打包成细胞数组以便 MATLAB 将输出参数传递给调用者。

使用 varargout 来返回可选的参数值有下面两种定义方式：

```
function varargout = myfun(vin1, vin2, ...)
function [vout1 vout2 ... varargout] = myfun(vin1, vin2, ...)
```

用第一种定义形式定义的函数，其函数体内创建了 varargout 细胞数组。此细胞数组的元素及其顺序决定了函数被调用时 MATLAB 如何为可选的返回值赋值。这种情况下 varargout 是函数定义行中等号左边唯一的变量，MATLAB 将 varargout{1}赋值给第一个返回参数，将 varargout{2} 赋值给第二个返回参数，依次类推。

用第二种定义形式定义的函数，除了 varargout，函数定义行中还有其他的返回参数。此时 MATLAB 先为调用函数时最左边返回参数赋值，然后按照顺序为 varargout 数组赋值。

例 8.27 倒序逐行输出 5 阶魔方矩阵的值。

byRow.m

```
function varargout = byRow(a)
varargout{1} = ' With VARARGOUT constructed by row ...';
```

```
for k = 1:5
    row = 5 - (k-1);                  % 翻转行的顺序
    varargout{k+1} = a(row,:);
end
```

varargout{1}为赋值为字符串常量。指定 4 个返回变量调用此函数，则 MATLAB 返回 varargout{1:4}，其中，varargout{2:4}返回矩阵的倒数三行。结果如下：

```
[text r1 r2 r3] = byRow(magic(5))
text = With VARARGOUT constructed by row ...
r1 =
    11    18    25     2     9
r2 =
    10    12    19    21     3
r3 =
     4     6    13    20    22
```

byRow 函数可以用 0~6 个返回参数来调用，如下面的调用方式都是正确的：

```
[text r1 r2 r3,r4,r5] = byRow(magic(5))     %6 个返回参数
byRow(magic(5))                              %不用返回参数
```

例 8.28 说明使用 varargin：下面的函数可以接受任意多个二维向量，然后用直线将以这些二维向量为坐标的点连接起来。

testvar.m

```
function testvar(varargin)
for k = 1:length(varargin)
    x(k) = varargin{k}(1);     %细胞矩阵索引
    y(k) = varargin{k}(2);
end
xmin = min(0,min(x));
ymin = min(0,min(y));
axis([xmin fix(max(x))+3 ymin fix(max(y))+3])
plot(x,y)
```

testvar 函数可以接受任意多的参数，比如，下面调用方式都是可以的：

```
testvar([2 3],[1 5],[4 8],[6 5],[4 2],[2 3])
testvar([-1 0],[3 -5],[4 2],[1 1])
```

由于 varargin 用一个细胞数组包含了所有的输入参数，所以，可以用细胞数组的索引来获取每个参数的值。例如：

```
y(n) = varargin{n}(2);
```

细胞数组索引有以下两个部分，用花括号{}的索引表示细胞数组细胞的索引，用圆括号()的索引表示某细胞的内容索引。像上面的代码，表达式{i}访问 varargin 细胞数值的第 i 个值，表达式(2)访问第 i 个细胞的第二个内容。

例 8.29 演示使用 varargout：输入参数的矩阵必须是 2 列，而行数可以任意，第 1 列为 x 坐标集合，第 2 列为 y 坐标集合。

testvar2.m

```
function [varargout] = testvar2(arrayin)
for k = 1:nargout
    varargout{k} = arrayin(k,:);     %细胞数组赋值
```

```
end
```

函数把每一行的坐标对分离成单独的[xi yi]，这些坐标值组成一个坐标向量。

调用 testvar2：

```
a = [1 2; 3 4; 5 6; 7 8; 9 0];

[p1, p2, p3, p4, p5] = testvar2(a)
p1 =
      1     2
p2 =
      3     4
p3 =
      5     6
p4 =
      7     8
p5 =
      9     0
```

varargin 和 varargout 必须出现在参数列表的最后，前面可以是任意个输入或输出参数。下面的 varargin 和 varargout 位置都是正确的：

```
function [out1,out2] = example1(a,b,varargin)
function [i,j,varargout] = example2(x1,y1,x2,y2,flag)
```

在嵌套函数中使用可选个参数时要特别注意，因为在嵌套函数中的 varargin，varargout，nargin 和 nargout 的具体含义可能会发生混乱。varargin 和 varargout 是变量，和其他变量一样，遵循 MATLAB 的变量作用域原则。另外，注意嵌套函数与所有外部函数共享工作空间。若 nargin 或 nargout 出现在嵌套函数中，则代表传递给这个函数的输入参数或输出参数的个数，不管这个函数是否是嵌套函数。若嵌套函数需要得到外部函数的 nargin 值和 nargout 值，可以将此作为参数传递。nargin 和 nargout 为函数，会记录被调用时的输入参数和输出参数。

若函数体中对输入参数进行修改，则需将此参数也列入输出参数，这样调用函数就能得到修改后的值。例如：

```
function [text, offset] = readText(filestart, offset)
```

调用 readText 函数时，读取某文件一行，则必须记录此次文件的偏移量 offset 以便下次调用时能获取开始读取的位置。但是每次 readText 函数调用结束后，offset 变量的值就从内存中清除了。为保存 offset 的值，采用上述调用方法，将 offset 作为返回值。

另外，MATLAB 提供了一个 inputParser 类来处理传递给 M 文件函数中不同类型的参数。

8.1.5　数据导入与导出

MATLAB 提供了将磁盘文件或剪贴板中的数据加载到工作空间的多种方法，称之为导入数据（Importing Data），同时也提供了多种将工作空间的变量保存到磁盘的方法，称之为导出数据（Exporting Data）。

选择不同的导入机制或导出机制取决于要传输的数据的格式，比如，文本文件、二进制文件和 JPEG 文件。MATLAB 内嵌了导入导出以下格式文件的功能。

- 二进制文件。

- 文本文件。
- 图形文件。
- 音频或视频文件。
- 电子数据表（Spreadsheets）。
- 剪贴板的数据。
- Internet 的信息。

除了 MATLAB 的导入函数外，还可以用工具箱来导入具有特定特点的数据，比如，可以使用 Database Toolbox 来导入关系数据库的数据。

1．使用导入向导（Import Wizard）

导入向导是 MATLAB 提供的一个图形交互界面，大大方便了数据的导入。若从文件导入数据，则执行【File】→【Import Data...】命令或下面命令行可以打开导入数据向导：

```
uiimport -file
```

若从剪贴板导入数据则执行【Edit】→【Paste to WorkSpace】命令，或用下面命令行：

```
uiimport –pastespecial
```

例 8.30 导入一个文本文件的数据到 MATLAB 工作空间。

文本文件 grades.txt 的内容如下：

```
John     85    90    95
Ann      90    92    98
Martin   100   95    97
Rob      77    86    93
```

打开导入向导对话框导入 grades.txt，第二步情况如图 8-3 所示。

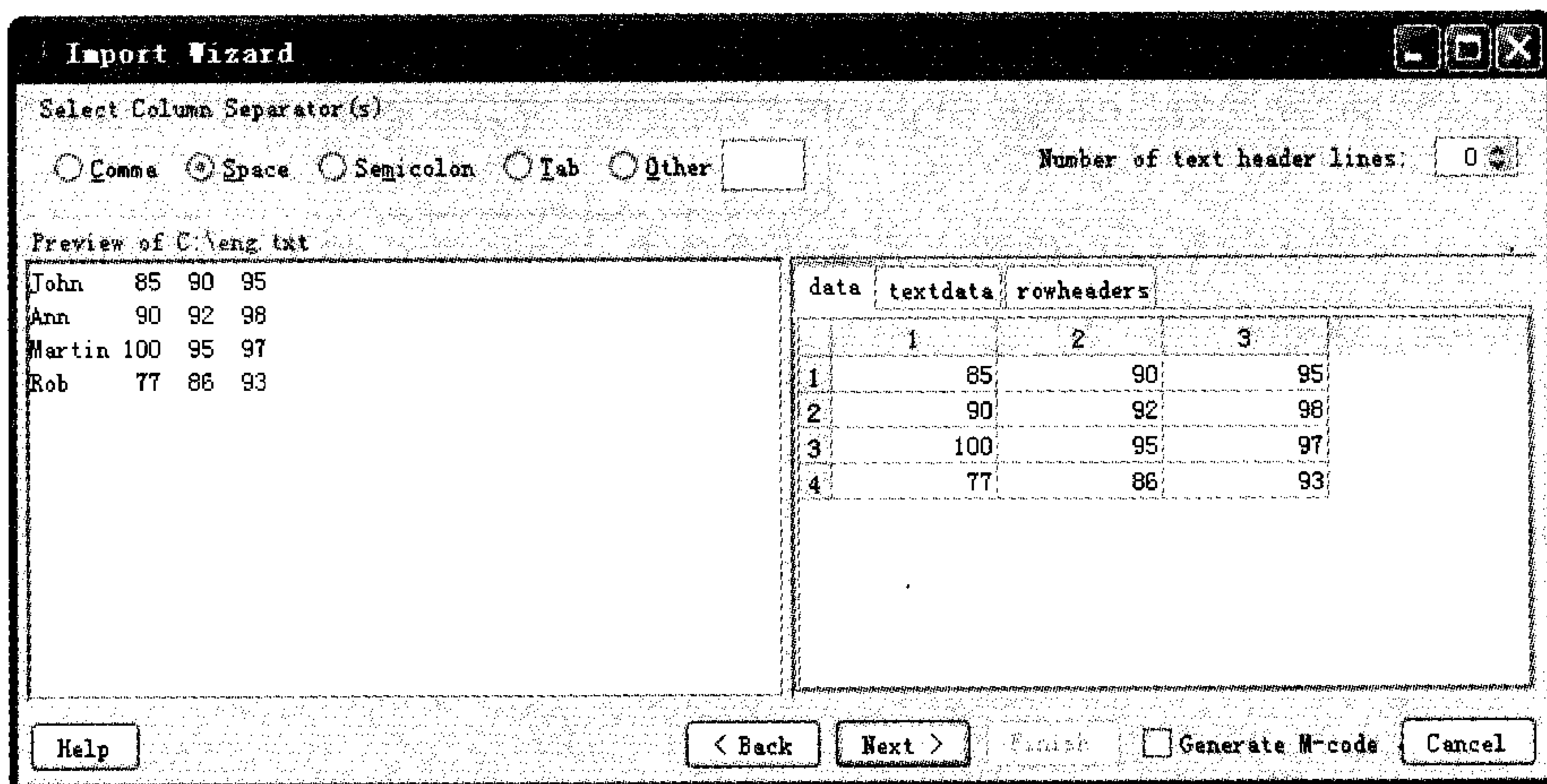


图 8-3 “ImportWizard”对话框

数据的行由文件的换行符来划分，而列的划分则由用户来指定。导入向导的左上角让用户选择列的分隔符。分隔符可以为逗号（Comma）、空格（Space）、制表符（Tab）、分号（Semicolon）和用户需要在右边编辑区输入自定义的分隔符（Other）。

界面的左边为文件内容预览，以文本格式显示，和用其他文本编辑器打开显示的内容与格式是相同的。界面的右边有 3 项内容：data、textdata 和 rowheaders/colheaders。其中，data 为文件中的数据，指数字部分；textdata 为文件中除数字部分外的数据，因为 MATLAB

默认导入数字数据；rowheader/colheader 是行名或列名，如本例中的第 1 列为每一行的行名，MATLAB 将其归为 textdata 中。在此向导中的第三步可以选择是否需要导入 textdata 和 rowheaders/colheaders，可以选择将数据的每一行以行名为变量名创建一个单独的行向量后导入，也可以仅导入用户指定的行。

在控制窗口使用下面命令也可以导入 grades.txt 的数据：

```
uiimport grades.txt      % grades.txt 要在当前目录下
```

然后，用 whos 命令查看工作空间的变量，与导入向导的结果一致：

```
whos
  Name              Size             Bytes  Class     Attributes

  data              4x3                 96  double
  rowheaders        4x1                272  cell
  textdata          4x1                272  cell
```

2．导入导出 MAT 文件

使用 save 函数可以将工作空间的变量导出为二进制或 ASCII 文件。可以保存工作空间中的所有变量或保存指定的某些变量。

将所有变量保存到 filename 文件中：

```
save filename
```

保存指定的变量：

```
save filename var1 var2 ... varN
```

在指定保存所需变量时，变量名称中可包含通配符"*"，下面的命令保存所有开头为 str 的变量：

```
save strinfo str*
```

用 whos –file strinfo 命令可以检查导入到此 MAT 文件的数据：

```
whos -file strinfo
  Name            Size                   Bytes  Class

  str2            1x15                      30  char array
  strarray        2x5                      678  cell array
  strlen          1x1                        8  double array
```

保存 MATLAB 结构体变量时，可以将结构体作为一个整体变量保存，也可以将结构体的每个域作为单独的变量保存，还可以以单独变量的形式保存某些指定的域。比如，对于结构体 S。

```
S.a = 12.7;   S.b = {'abc', [4 5; 6 7]};   S.c = 'Hello!';
```

保存整个结构体：

```
save newstruct.mat S;
whos -file newstruct
  Name      Size                   Bytes  Class

  S         1x1                      550  struct array
```

使用-struct 选项单独保存每个域为独立的变量：

```
save newstruct.mat -struct S;
whos -file newstruct
  Name      Size                   Bytes  Class
```

```
  a           1x1                       8  double array
  b           1x2                     158  cell array
  c           1x6                      12  char array
```

或保存指定的域为独立的变量：

```
save newstruct.mat -struct S a c;

whos -file newstruct
  Name        Size                  Bytes  Class

  a           1x1                       8  double array
  c           1x6                      12  char array
```

load 函数可将磁盘上的二进制文件或 ASCII 文件导入到 MATLAB 工作空间：

```
load filename
```

或导入指定的变量（同样也可以使用通配符“*”）：

```
load filename var1 var2 ... varN
```

也可以将 MAT 文件中的数据导入到一个结构体中：

```
S = load('mydata.mat')
```

3．导入导出图形文件

使用 imread 函数可以将图形文件导入到 MATLAB 工作空间。imread 函数支持标准文件格式的图形文件，包括 TIFF（Tagged Image File Format）、GIF（Graphics Interchange Format）、JPEG（Joint Photographic Experts Group）以及 PNG（Portable Network Graphics）格式。下面命令将 JPEG 格式的图形数据读取到 MATLAB 工作空间，用数组 I 存储：

```
I = imread('myphoto.jpg');
```

imread 用多维数组来表示图像数据，维数取决于图像的格式。比如，使用三维数组代表 RGB 颜色图像：

```
whos I
  Name        Size                  Bytes  Class

  I        650x600x3              1170000  uint8 array
```

imwrite 函数可从 MATLAB 工作空间中导出标准格式的图形文件，支持的格式与 imread 相同。下面命令将 MATLAB 工作空间中的多维数组数据 I 读取到 TIFF 格式的文件中：

```
imwrite(I, 'my_graphics_file.tif','tif');
```

4．导入导出音频视频文件

MATLAB 有很多函数可以查询包含音频或视频数据的文件信息，如 mmfileinfo 函数。同时，MATLAB 提供了很多导入音频视频数据到工作空间的函数，可以从文件中导入，也可以利用输入设备录制，如用麦克风。

导入音频视频的函数有 auread、aviread 和 wavread，可分别读取声音文件、AVI 视频和 WAVE 声音。

MATLAB 中导出音频的函数有 auwrite 和 wavwrite，分别可导出声音为 AU 和 WAV 格式文件。而导出视频文件就复杂一些，需要用 avifile 函数创建 avifile 对象，然后利用 AVI 文件对象的方法和属性来控制导出过程。如在 MATLAB 中，可把一系列的图形保存为可播

放的电影，然后导出为 MAT 文件。

例 8.31　创建 AVI 文件：将一系列图像保存为 AVI 格式的电影。

CreateAVI.m

```
%1 创建 AVI 文件对象：
aviobj = avifile('mymovie.avi','fps',5);

%2 抓图，存进 AVI 文件中：
for k=1:25
    h = plot(fft(eye(k+16)));
    set(h,'EraseMode','xor');
    axis equal;
    frame = getframe(gca);
    aviobj = addframe(aviobj,frame);
end

%3 关闭 AVI 文件：
aviobj = close(aviobj);
```

5．**导入导出电子数据表**（Spreadsheets）

MATLAB 支持微软的 Excel 电子数据表和 Lotus 的 1-2-3 电子数据表。由于前者用户较多，在此只介绍 Excel 的导入导出。

使用 xlswrite 函数可将矩阵导出为 Excel 表。如 *d* 为包含文字和数字混合数据的矩阵：

```
d = {'Time', 'Temp'; 12 98; 13 99; 14 97}
d =
    'Time'    'Temp'
    [  12]    [  98]
    [  13]    [  99]
    [  14]    [  97]
```

将 *d* 导出为名为 tempdata.xls 的 XLS 文件，标签名为 Temperatures，数据从 E1 位置开始写：

```
xlswrite('tempdata.xls', d, 'Temperatures', 'E1');
```

xlsread 函数可将 Excel 文件的数据导入到 MATLAB 工作空间。例如，将上面导入的 XLS 文件中的数据导入：

```
ndata = xlsread('tempdata.xls', 'Temperatures')
ndata =
      12     98
      13     99
      14     97
```

MATLAB 仅仅导入数字信息，而忽略了文字信息。若要将文字数据也导入，需用两个返回参数：

```
[ndata, headertext] = xlsread('tempdata.xls', 'Temperatures')
headertext =
      'Time'     'Temp'
ndata =
      12     98
      13     99
```

14 97

6．导入导出 Internet 数据

MATLAB 提供了与 Internet 交互的函数，支持一些基本的协议，如 FTP、SMTP 和 HTTP，可以发送 E-mail、从 Internet 下载文件、压缩文件（Zip）、解压缩文件（Unzip)、连接 FTP 站点进行远程文件操作。

urlread 函数可将网页中的数据导入到 MATLAB 基本空间，urlwrite 函数可将网页保存为文件。如下面命令将查询“Simulink”结果导入为变量 s：

```
s = urlread('http://www.google.com/search','get',{'q','Simulink'})
```

将 mathworks 主页保存为网页文件：

```
urlwrite('http://www.mathworks.com/', mathworks.html');
```

zip 和 unzip 函数可压缩、解压缩文件，如：

```
zip('simulink_matches.zip','contains_simulink.html');
```

sendmail 函数可以发送邮件，但首先需要用 setpref 函数定义好相关设置，包括 E-mail 地址和 SMTP 服务器，如：

```
setpref('Internet','E_mail','saunala@126.com');
setpref('Internet', 'SMTP_Server', 'mail.server.network');
```

下面就可以使用 sendmail 函数发送 E-mail 了，至少包括两个参数，即接收者地址和邮件标题，如：

```
sendmail('recepient@someserver.com', 'Hello From MATLAB!');
```

MATLAB 可以连接 FTP 服务器进行远程操作，如下面的例子。

例 8.32 读取 MathWorks FTP 服务器上的 pub/pentium/Moler_1.txt 文件(此例可运行，FTP 服务器和目录都是有效的)。

readFTP.m

```
tmw=ftp('ftp.mathworks.com');   %连接 FTP server
cd(tmw,'pub');                  %切换到 pub 路径下,需要有 tmw 参数
cd(tmw,'pentium');              %从 pub 路径切换到 pentium 目录
dir(tmw) ;                      %列出 pentium 目录下内容
mget(tmw,'Moler_1.txt');        %将 Moler_1.txt 文件导入到 MATLAB 当前路径
close(tmw);                     %关闭 FTP 路径连接
```

8.1.6 实例分析

本节将介绍大量的实例，以便对前面介绍的内容有更好的理解。

例 8.33 分别建立脚本文件和函数文件，将华氏温度 F 转换为摄氏温度 C。

此例比较简单，只需知道华氏温度和摄氏温度的换算关系就不难编写出正确的程序。

程序 1：首先建立脚本文件并以文件名 f2c1.m 存盘：

```
f=input('Input Fahrenheit temperature：');
c=5*(f-32)/9
```

然后在 MATLAB 的控制窗口中输入 f2c1，将会执行该脚本文件，执行情况如下：

```
Input Fahrenheit temperature：73
c =
 22.7778
```

程序 2：首先建立函数文件 f2c2.m：

```
function c=f2c(f)
c=5*(f-32)/9;
```

然后在 MATLAB 的控制窗口调用该函数文件：

```
y=input('Input Fahrenheit temperature: ');
x=f2c2(y)
```

输出情况如下：

```
Input Fahrenheit temperature: 70
x =
   21.1111
```

例 8.34　输入 x，y 的值，并将它们的值互换后输出。

考察 input 函数的用法，程序如下：

inputsam.m

```
x=input('Input x please.');
y=input('Input y please.');
z=x;
x=y;
y=z;
disp(x);
disp(y);
```

例 8.35　求一元二次方程 $ax^2+bx+c=0$ 的根。

用 input 获得系数参数的值，然后用求根公式求值，程序如下：

equation.m

```
a=input('a=?');
b=input('b=?');
c=input('c=?');
d=b*b-4*a*c;
x=[(-b+sqrt(d))/(2*a),(-b-sqrt(d))/(2*a)];
disp(['x1=',num2str(x(1)),',x2=',num2str(x(2))]);
```

例 8.36　计算分段函数的值 y。

分段函数为 $\begin{cases} \dfrac{x+\sqrt{\pi}}{\mathrm{e}+x^2}, & x \leqslant 0 \\ \log(x+\sqrt{1+x^2})/2, & x>0 \end{cases}$

程序如下：

equation2.m

```
x=input('请输入 x 的值:');
if x<=0
y= (x+sqrt(pi))/(e+x^2)
else
y=log(x+sqrt(1+x*x))/2
end
```

例 8.37　输入一个字符，若为大写字母，则输出其对应的小写字母；若为小写字母，则输出其对应的大写字母；若为数字字符则输出其对应的数值，若为其他字符则原样输出。

本例是多分支的情况，可利用 if-else 语句完成。

character.m

```
c=input('请输入一个字符','s');
if c>='A' & c<='Z'
disp(setstr(abs(c)+abs('a')-abs('A')));
elseif c>='a'& c<='z'
    disp(setstr(abs(c)- abs('a')+abs('A')));
elseif c>='0'& c<='9'
    disp(abs(c)-abs('0'));
else
    disp(c);
end
```

例 8.38　某商场对顾客所购买的商品实行打折销售，标准如下（商品价格用 price 来表示）。

price<200	没有折扣
200≤price<500	3%折扣
500≤price<1000	5%折扣
1000≤price<2500	8%折扣
2500≤price<5000	10%折扣
5000≤price	14%折扣

输入所售商品的价格，求其实际销售价格。

本例分的情况较多，可用 switch 语句实现，程序如下：

switchexa.m

```
price=input('请输入商品价格');
switch fix(price/100)
 case {0,1}                %价格小于 200
     rate=0;
 case {2,3,4}              %价格大于等于 200 但小于 500
     rate=3/100;
 case num2cell(5:9)        %价格大于等于 500 但小于 1000
     rate=5/100;
 case num2cell(10:24)      %价格大于等于 1000 但小于 2500
     rate=8/100;
 case num2cell(25:49)      %价格大于等于 2500 但小于 5000
     rate=10/100;
 otherwise                 %价格大于等于 5000
     rate=14/100;
end
price=price*(1-rate)       %输出商品实际销售价格
```

例 8.39　矩阵乘法运算，要求两矩阵的维数相容，否则会出错。先求两矩阵的乘积，若出错，则自动转去求两矩阵的点乘。

本例练习 try-catch 语句，程序如下：

try_catch.m

```
A=[1,2,3;4,5,6]; B=[7,8,9;10,11,12];
try
 C=A*B;
 catch
   C=A.*B
end
lasterr                    %显示出错原因
```

运行此脚本，结果如下：

```
ans =
    Error using ==> mtimes
    Inner matrix dimensions must agree.
```

例 8.40 一个三位整数，各位数字的立方和等于该数本身，则称该数为水仙花数。输出全部水仙花数。

程序如下：

daffodil.m

```
for m=100:999
   m1=fix(m/100);              %求 m 的百位数字
   m2=rem(fix(m/10),10);       %求 m 的十位数字
   m3=rem(m,10);               %求 m 的个位数字
   if m==m1*m1*m1+m2*m2*m2+m3*m3*m3
      disp(m)
   end
end
```

例 8.41 已知 $y=1+\frac{1}{3}+\frac{1}{5}+\cdots+\frac{1}{2n\text{-}1}$，当 n=100 时，求 y 的值。

程序如下：

arrange.m

```
y=0;
n=100;
for i=1:n
  y=y+1/(2*i-1);
end
y
```

在实际 MATLAB 编程中，采用循环语句会降低其执行速度，所以前面的程序通常由下面的程序来代替：

```
n=100;
i=1:2:2*n-1;
y=sum(1./i);
```

例 8.42 写出下列程序的执行结果。

loopsam.m

```
s=0;
a=[12,13,14;15,16,17;18,19,20;21,22,23];
for k=a
    s=s+k;
end
disp(s');
```

for 语句更一般的格式如下：

```
for  循环变量=矩阵表达式，
    循环体语句
end
```

执行过程是依次将矩阵的各列元素赋给循环变量，然后执行循环体语句，直至各列元素处理完毕。故上述代码中 for 循环中即是将矩阵的各行分别累加，最后结果为列向量。

本例执行结果如下：

```
39      48      57      66
```

例 8.43 从键盘输入若干个数，当输入 0 时结束输入，求这些数的平均值和它们之和。

可用 while 语句来实现，结束条件为输入字符为 0，程序如下：

whileexa.m

```
Sum=0;
Cnt=0;
Val=input('Enter a number (end in 0):');
while (Val~=0)
    Sum=Sum+Val;
    Cnt=Cnt+1;
    Val=input('Enter a number (end in 0):');
end
if (Cnt > 0)
    Sum
    Mean=Sum/Cnt
end
```

例 8.44 求任意两个数的最大公约数和最小公倍数。

程序如下：

GCD.m

```
A1= input('输入两个非零数  第一个:');
A2= input('输入两个非零数  第二个:');
a=max(A1,A2);
b=min(A1,A2);
while(b~=0)
   r=rem(a,b);
   a=b;                          %利用代数学中的辗转相除法
   b=r;
end
disp(a);
disp(A1*A2/a);                   % A，B 两数的最小公倍数为 A×B / (A 和 B 的最大公约数)
```

完成一个功能，选择正确高效率的算法是关键。

例 8.45　求[100，200]之间第一个能被 21 整除的整数。

程序如下：

loopsam2.m

```
    for n=100:200
    if rem(n,21)~=0
          continue
    end
    n
    break
end
```

本例综合考查 continue 和 break 的使用方法，以及二者的区别。

例 8.46　若一个数等于它的各个真因子之和，则称该数为完数，如 6=1+2+3，所以 6 是完数。求[1,500]之间的全部完数。

程序如下：

wanshu.m

```
    for m=1:500
    s=0;
    for k=1:m/2
      if rem(m,k)==0
      s=s+k;
      end
    end
    if m==s
        disp(m);
    end
end
```

执行结果如下：

```
         6
        28
       496
```

例 8.47　鸡兔同笼问题：鸡和兔子关在一个笼子里，已知共有 36 个头， 100 只脚，求笼内关了多少只兔子和多少只鸡？

使用 MATLAB 来解一般的方程，程序如下：

chirabbit.m

```
for chicken=1:36
     if rem(100-chicken*2, 4)==0 & (chicken+(100-chicken*2)/4)==36
          break
     end
     chicken=chicken+1;
end
chicken
rabbit=(100-2*chicken)/4
```

运行结果如下：

```
chicken =
          22
rabbit =
          14
```

例 8.48 利用函数文件，实现直角坐标（x，y）与极坐标（ρ，θ）之间的转换。

函数文件 tran.m：

```
function [rho,theta]=tran(x,y)
rho=sqrt(x*x+y*y);
theta=atan(y/x);
```

调用 tran.m 的脚本文件 main.m：

```
x=input('Please input x=:');
y=input('Please input y=:');
[rho,theta]=tran(x,y);
rho
theta
```

例 8.49 利用函数的递归调用，求 $n!$。

$n!$本身就是以递归的形式定义的，求 $n!$需要求$(n-1)!$，这时可采用递归调用。递归调用函数文件 factor.m 如下：

factor.m

```
function f=factor(n)
if n<=1
  f=1;
else
  f=factor(n-1)*n;     %递归调用求(n-1)!
end
```

例 8.50 nargin 用法示例。

函数文件 charray.m

```
function fout=charray(a,b,c)
if nargin==1
  fout=a;
elseif nargin==2
  fout=a+b;
elseif nargin==3
  fout=(a*b*c)/2;
end
```

脚本文件 mydemo.m

```
x=[1:3];
y=[1;2;3];
a1 = charray(x)
a2 = charray (x,y')
a3 = charray (x,y,3)
```

上例用不同的参数个数调用了 charray 函数。执行结果如下：

```
a1 =
```

```
        1     2     3
a2 =
        2     4     6
a3 =
       21
```

例 8.51　Fibonaccii 数列定义如下：

$$\begin{cases} f_1=1 \\ f_2=1 \\ f_n=f_{n-1}+f_{n-2} \quad (n>2) \end{cases}$$，求 Fibonaccii 数列的第 20 项 f_{20}。

程序如下：

fibonaccii.m

```
function f=fibonaccii(n)
if n==0||n==1
    f=1;
else
    f=fibonaccii(n-1)+fibonaccii(n-2);
end
```

例 8.52　循环实现计算变量的 sin、cos 和 tan 值。

将 sin、cos 和 tan 放在一个细胞数组中，用 eval 即可实现，程序如下：

evalexa.m

```
CEM={'cos','sin','tan'};
for k=1:3
    theta=pi*k/12;
    y2(1,k)=eval([CEM{k},'(',num2str(theta),')']);
end
```

得出结果如下：

```
y2 =
    0.9659    0.5000    1.0000
```

例 8.53　串演算函数 eval 的应用。

程序如下：

evalexa2.m

```
t=0:0.01*pi:2*pi;
P=[1,2,4,10];
for n=1:4
    eval(['T=P(n);'])   %T 依次等于数组 P 中的数值
    y=sin(T*t);
    subplot(2,2,n)      %按顺序选择 4 个子图进行绘图
    plot(t,y)
    title(['y=sin(',num2str(T),'*t)'])
end
```

输出结果如下：

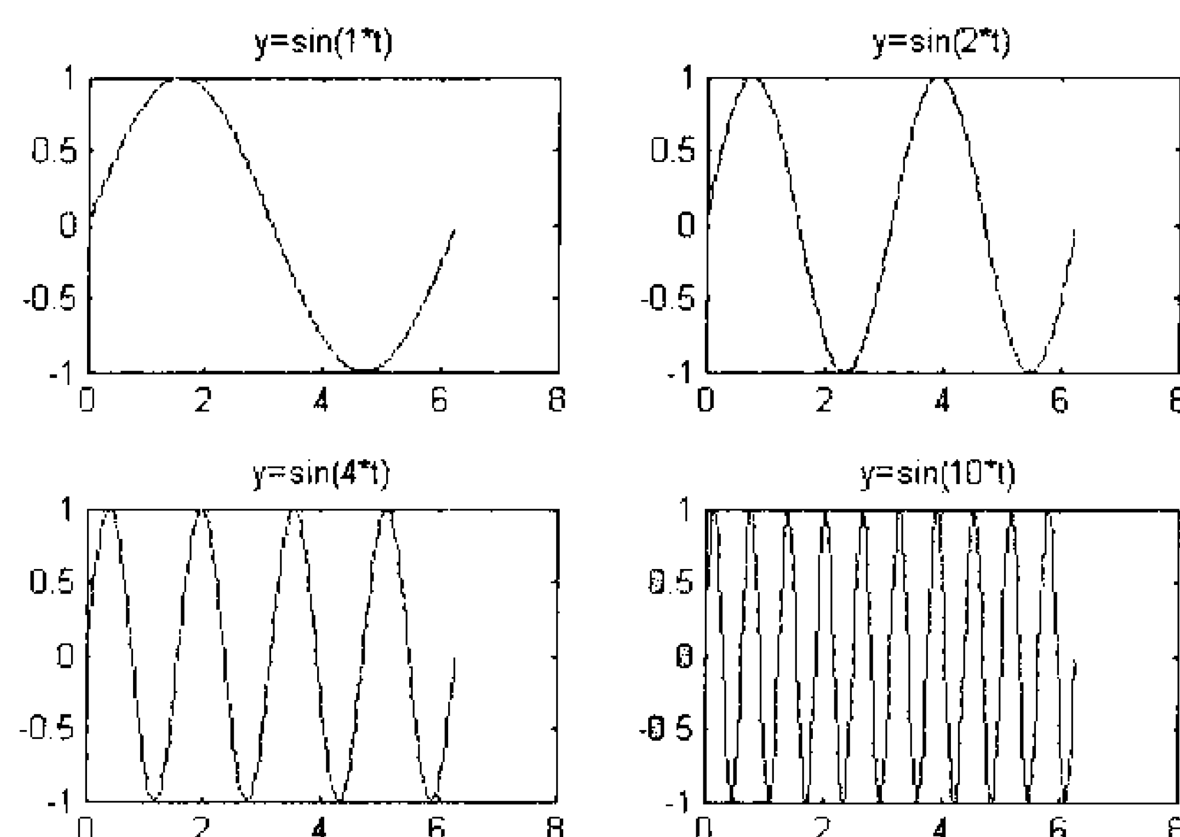


图 8-4 evalexa.m 的执行结果

例 8.54 计算（$a+b$）^{k}和（$a-b$）^{k}的值。其中 k=1，2，…，n.

可用脚本文件结合函数文件来实现——计算单次的（$a+b$）^{k}和（$a-b$）^{k}使用函数文件来完成，然后脚本文件调用函数文件来实现基本功能。

建立函数文件 mypower.m：

```
function [x,y] = mypower(a,b,n)
%compute (a+b)^n and (a-b)^n
x = (a+b)^n;
y = (a-b)^n;
```

建立调用上述函数文件的脚本文件 power.m：

```
a = input('Please input a= : ');
b = input('Please input b= : ');
x = zeros(1,10);
x = zeros(1,10);
for k=1:10
        [x(k),y(k)] = mypower(a,b,k);
end
x,y
```

输入脚本名称运行：

```
power
Please input a= :2
Please input b= :4
```

得出结果如下：

```
x =
     6     36    216    1296    7776    46656    279936    1679616    10077696    60466176
y =
    -2    4     -8     16     -32     64     -128     256     -512     1024
```

例 8.55 绘制$y=1-\dfrac{1}{\beta}e^{-\xi t}\sin(\beta t+\theta)$，$\xi=0.2,0.4,0.6,0.8$，$t=[0,18]$的曲线。

此例练习 MATLAB 的数学计算和绘图，程序如下：

huitu.m

```
t=[0:0.1:18]';
for x=0.2:0.2:0.8
    b=sqrt([1-x^2]);
    z=atan(b/x);
```

```
        y1=-t*x; y2=t*b+z;
        y=1-exp(y1).*sin(y2)/b;
        plot(t,y), hold on
    end
    xlabel('t（秒）'),ylabel('y')
    title('二阶系统阶跃响应')
    text(3.3,0.9,'{\xi}=0.8')
    text(4.3,1.4,'{\xi}=0.2')
```

绘图结果如下：

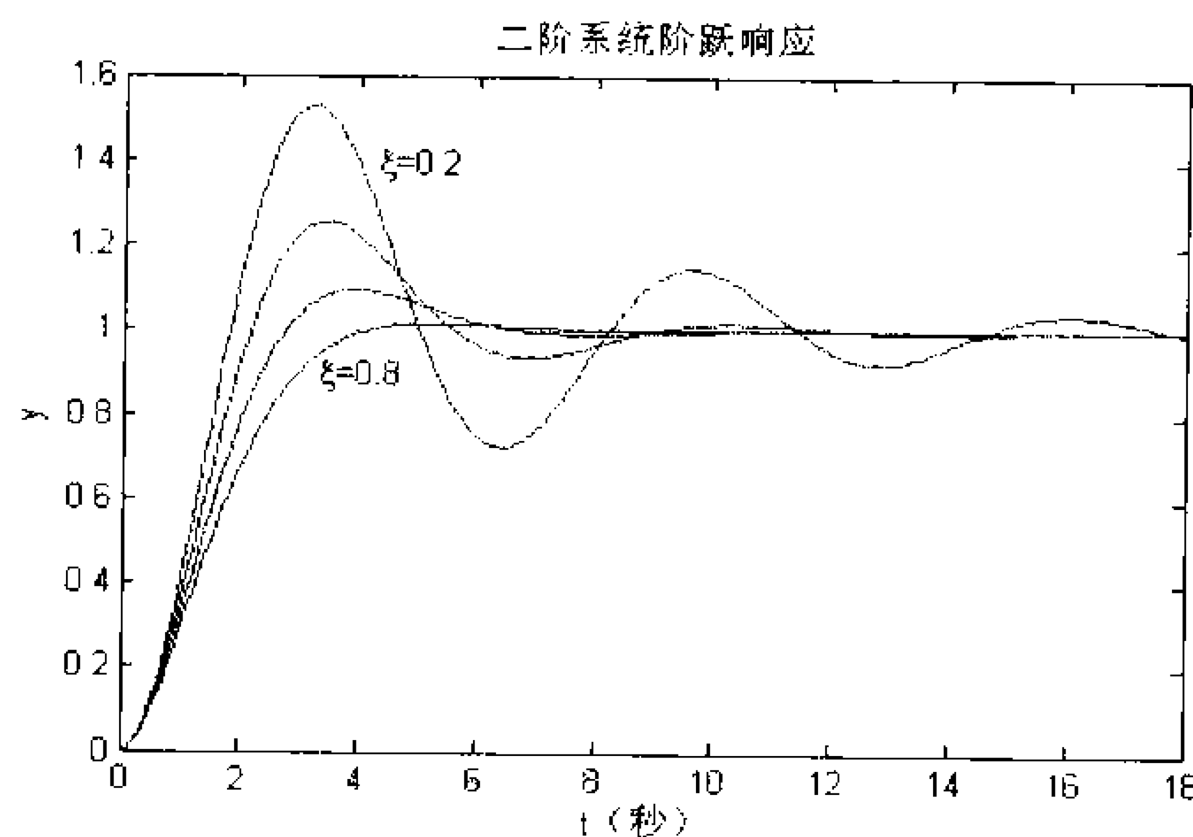


图 8-5　二阶系统阶跃响应结果图

8.2　M 文件编程的技巧

本节将分类讲解一些 MATLAB 编程的技巧，每个分类的提示都比较简明扼要，有些也已经在前面涉及。

8.2.1　命令和函数的语法

（1）获得 MATLAB 函数和命令的语法帮助可直接在命令行输入：

```
help syntax
```

（2）在控制窗口可以用命令方式和函数方式来调用函数，如：

```
functionname arg1 arg2 arg3                 % 命令语法
functionname('arg1','arg2','arg3')          % 函数语法
```

（3）若某个命令太长，一行写不下，或者便于阅读和理解采用多行来写，则在一行的末尾写续行符“…”，如：

```
sprintf ('Example %d shows a command coded on %d lines.\n', ...
                exampleNumber, ...
                numberOfLines)
```

但是，一个字符串必须在一行内完成，内部不能断行，比如：

```
disp 'This statement attempts to continue a string ...
      to another line, resulting in an error.'
```

在控制窗口输入后，MATLAB 会报错。

（4）可以用以下方法来调用以前调用过的命令。

- 在控制窗口输入命令处用上箭头和下箭头来选择要执行的命令。
- 若已知命令的前几个字母，可以在控制窗口输入命令处先输入已知字母，然后结合上下箭头来查找。这种方法会加快查找速度。
- 直接在历史窗口中查找。

（5）产生很大的中间结果时，最好不要在屏幕上显示，即在命令后以分号结束，如：

```
A = magic(100);      % 创建魔方矩阵，但不显示
```

（6）在 MATLAB 的控制窗口也可以运行 DOS 命令，需要在命令前加操作符“!”。在命令后加&则表示结果在 Windows 窗口中显示，否则在 MATLAB 的控制窗口显示。

如在 MATLAB 控制窗口输入检测磁盘的 DOS 命令 Chkdsk（在前面加!）：

```
!chkdsk
```

则在 MATLAB 控制窗口中显示：

```
文件系统的类型是 NTFS。

警告! 没有指定  F 参数
用只读模式运行 CHKDSK。

CHKDSK 正在校验文件(3 的阶段 1)...
。。。
总共有      4883728 KB 磁盘空间。
17985 个文件中有     4122224 KB。
2124 个索引      12936 KB。
不正确扇区          0 KB。
系统正在使用       55684 KB。
日志文件占用了       26480 KB。
磁盘上      692884 KB 可用。

每个分配单元中有        4096 字节。
磁盘上共有      1220932 个分配单元。
磁盘上有      173221 个可用的分配单元。
```

若在命令后添加&：

```
!chkdsk&
```

则会触发 Windows 的 DOS 窗口，并执行 chkdsk 命令。

8.2.2 获取帮助

若为子函数写了帮助，比如 myfun.m 中的子函数 mysubfun，可以这样获取帮助：

```
help myfun>mysubfun
```

显示私有函数的帮助，需在函数名前加 private/，如获取私有函数 myprivfun 的帮助：

```
help private/myprivfun
```

获取某类的方法帮助的方法如下：

```
help classname/methodname
```

其中，methodname.m 在子目录@classname 中。比如，为 polynom 类写了绘图 plot 方法（plot 方法在@polynom/plot.m 中定义）。可以输入：

```
help polynom/plot
```

用同样的方法可获得重载函数的帮助，如获取 matlab/iofun/@serial 目录下 eq 函数的帮

助：

```
help serial/eq
```

8.2.3 M 文件函数

（1）函数名使用小写字体。

由于有些系统是区分大小写的，故 M 文件函数名均采用小写字体有利于程序在不同系统间的移植。

（2）获取正在执行的 M 文件函数的名字，可在 M 文件函数中使用 mfilename 函数。若同时想获得函数的路径，则使用 mfilename('fullpath')。

（3）可用下面的方法来获取某函数执行时涉及的 M 文件。

- 用 clear 函数清理内存中所有的函数（除用 mlock 锁定的之外，可用 munlock 来解锁，然后重复第一步）。
- 执行要检测的函数，函数的参数很重要，同一个函数使用不同的参数可能会得到不同的结果。
- 调用 inmem 显示函数执行时使用的 M 文件，若想查看同时使用的 MEX 文件，则要使用另外的输出参数，比如：

```
[mfiles, mexfiles] = inmem
```

（4）查看函数更多的依赖细节信息可用 depfun 函数，对于 M 文件，可返回所依赖的内嵌函数和类的信息。

（5）M 函数可以有多个输入参数，也可以没有。

（6）函数可以少于函数规定的输入输出参数个数，但不能多于函数 M 文件中所规定的输入和输出变量数目。如果输入和输出参数数目多于函数 M 文件中 function 语句所规定的数目，则调用时自动返回一个错误。

（7）函数名最好和 M 文件名相同。

（8）M 函数支持函数间的调用。

若一个 M 文件包含一个以上的函数，则一个为主函数，其他为子函数。主函数要排在 M 文件最前面，而子函数可以任意排列。子函数只能被本 M 文件中的主函数和其他子函数调用，在此 M 文件外，子函数是不可见的。

（9）函数 M 文件可调用脚本文件。

此时被调用的脚本文件共享此 M 文件的工作空间，而不是共享 MATLAB 的工作空间。从函数 M 文件内调用的脚本文件不必用调用函数编译到内存。函数每调用一次，它们就被打开和解释。因此，从函数 M 文件内调用脚本文件减慢了函数的执行。

（10）一般脚本文件作为主文件，而用函数文件完成各种不同的子功能，再在脚本文件中调用完成整个程序的目标。这样增加了函数文件代码的重用性，在其他程序中若完成同样或类似的功能，可直接调用这些函数文件。

（11）调用函数时可使用句柄，而使用句柄有如下几个优点。

- 只要函数句柄可以正确创建，不论函数是否在当前搜索路径上，是否是子函数、私有函数都可以正确执行。

- 使用函数句柄，重载函数仍可以正确运行。
- 使用函数句柄可以省去搜索路径的麻烦，加快计算速度。

8.2.4 程序开发

（1）最好采用结构化编程思想，将一个程序分解成若干个相互独立的小问题。每个问题用各自的函数解决。这样每个函数变得简短，意图明确，便于阅读和调试。

（2）使用伪码（Pseudo-Code）。

在程序设计最初用自己语言写程序草稿时，使用伪码比较有用。从伪码可以程序的思路与算法思想，便于得出结论，便于检查和修改，在程序开发下一步也容易转换为编程语言。

如 8.1.6 节中判断一个三位数 N 是否为水仙花数的伪码可以写为：

```
for N=100:999
    计算 N 的百位数 a
    计算 N 的十位数 b
    计算 N 的个位数 c
    if a 的立方 ＋ b 的立方 ＋ c 的立方 ＝＝ N
      disp(N)
    end
 end
```

（3）基本的编程编码习惯（仅为建议，而非必要）。

①函数名和变量名最好有意义，便于理解，如矩形的长定义为 Length，计算自然数的阶乘的函数名定义为 factorial()。

②同一个 M 文件中的子函数按照字母排序编写，有利于查找。

③子函数最好编写帮助，不仅知道函数用途，而且便于子函数之间的区分。

④一行的代码不要多于 80 列，否则，打印出来后难以阅读。

⑤图像句柄（Handle Graphics）的特性和值使用全名，名字缩写也允许使用，但会使代码难以阅读，在以后的 MATLAB 版本中或许不再支持。

（4）编写必要的注释，会增加程序的可读性。可为关键算法、不明显的段、主要程序段等加注释。特别重要的注释可以用下面的方式：.

```
%-----------------------------------------------
% 这个函数的功能是完成……
%-----------------------------------------------
```

（5）程序编写按步来，并逐段地运行查看结果，这样比程序全部写完后再运行更便于查找错误。同样，修改程序时也一点一点修改，并逐步运行查看，容易查找出出错的代码。

（6）若一个函数仅仅为另外一个函数所调用，则最好把这两个函数放在同一个 M 文件，并把被调用的函数编写成子函数。

（7）有两种方法可以获得文件的大小，例子如下：

```
-- 方法 #1 --                    -- 方法 #2 --
s = dir('myfile.dat');          fid = fopen('myfile.dat');
filesize = s.bytes                fseek(fid, 0, 'eof');
                                      filesize = ftell(fid)
```

```
fclose(fid);
```

8.2.5 变量

（1）命名一个变量后要确认变量名是合法的，合法的变量名长度可由下列命令来获取：

```
N = namelengthmax
N =
    63
```

即此版本 MATLAB 能识别的变量长度为 63，另外可由 isvarname 来判断一个变量名是否合法：

```
isvarname 8thColumn
ans =
     0
```

即 8thColumn 不能用作变量名，因为用了数字开头。

（2）变量名尽量不要和函数名相同，否则将不能正确调用此函数。同样，变量不能与 MATLAB 的预留关键字重名，用 iskeyword 命令可查询 MATLAB 的关键字。

（3）MATLAB 使用字母 i 和 j 代表复数的虚部，所以在复数计算中避免使用 i 和 j 来作为变量名。

（4）MATLAB 脚本中的变量存储在调用它的文件的基本空间中，若在命令行中调用脚本，则与 MATLAB 共享基本工作空间，若被函数调用，则共享此函数的工作空间。所以，脚本中的变量尽量与其他函数或 MATLAB 环境中不同名，否则在调用脚本时，会不经意地修改函数或 MATLAB 中的变量而不被察觉。

（5）若使用 MATLAB 的静态变量，则可用 persistent 来定义永久变量。变量在一个函数中被声明为 persistent 型后，在函数被调用期间，会一直保留在内存中。不像全局变量，永久变量仅在声明它的函数中可见。

（6）慎重使用全局变量。全局变量在所有的函数和 MATLAB 基本空间中都可见，全局变量使用越多，变量重名的机会就越大。故在某处定义的全局变量在另外的地方修改不容易察觉，由此导致的错误难以调试和追踪。

（7）使用 input 可以使程序和用户交互，让用户输入数值给变量赋值。

（8）变量名应该尽量反映其用途，小范围使用的变量应使用短的变量名。

（9）结构体最好用大写字母开头。与 C 语言类似，这样可以区分结构体和其他变量。

8.2.6 字符串

（1）创建一个字符串常用的方法是由多个字符串连接而成，连接多个字符串可以使用连接符（[]）或 sprintf 函数（与 C 语言类似），下面的第 2 行和第 3 行分别使用了这两种方法，得到的结果是一样的。

```
numChars = 28;
s = ['There are ' int2str(numChars) ' characters here']
s = sprintf('There are %d characters here\n', numChars)
```

在连接字符串时，sprintf 比[]适合，因为 sprintf 可读性好，特别是操作复杂的字符串。

执行速度也快。

（2）用细胞数组而不是字符数组来存储字符串。

在字符串长度不相同时，若存储在字符数组，短的字符串需要填补空格来使每个字符串与最长的字符串长度相同，因为字符数组要求每个字符串长度相同。存储为细胞矩阵时就不必填补空格，如：

```
Record = {'Allison Jones'; 'Development'; 'Phoenix'};
```

（3）可在标准字符数组和细胞数组之间转换。

cellstr 可将一字符数组转化为细胞数组：

```
charRecord = ['Allison Jones'; 'Development  '; 'Phoenix      '];
cellRecord = cellstr(charRecord);
```

另外，MATLAB 的一些字符串操作可用于字符数组，有的可用于细胞数组，有的二者都适用，如 strcmp 可以比较字符数组和细胞数组：

```
cellRecord2 = {'Brian Lewis'; 'Development'; 'Albuquerque'};
strcmp(charRecord, cellRecord2)
ans =
     0
     1
     0
```

（4）对于常规的 MATLAB 表达式，MATLAB 提供了通用的查找和替换方法，可以查找替换字符串中的字符和短语。请查询 regexp、regexpi 和 regexprep 的帮助，在此不进行详述。

（5）eval 和 feval 指令。eval 和 feval 指令都是用来执行字符串所代表的函数，优点在于可以在运行中修改所执行的指令或参数，提高计算的灵活性。

eval 指令的基本语法如下：

①eval 为执行 CEM 指定的运算。

```
y=eval('CEM')
```

②执行 CEM1 指定的运算，失败则执行 CEM2。

```
y=eval('CEM1', 'CEM2')
```

③调用 CEM 代表的函数文件，输出计算结果。

```
[y1,y2,...]=eval('CEM')
```

feval 指令的基本语法如下：

```
[y1,y2,...]=feval('FUN',arg1,arg2,...)
```

上述指令调用函数 FUN，输入参数为 arg1，arg2，…，输出计算结果。

二者区别在于：

①eval 所执行的字符串应该包括函数名、输入参数，甚至输出参数。

②feval 的 FUN 字符串仅为函数名，输入参数由 arg1，arg2 给出。

例如，下面命令完成同样的操作：

```
eval('s=sin(pi)' )
s=eval('sin(pi)')
s=feval('sin',pi)
```

8.2.7 表达式求值

（1）限制使用求值函数 eval。

eval 函数在某些场合特别有用，但是最好限制其使用。原因之一是使用 eval 的代码难以阅读，同时难以调试。另外，eval 有时不能由 MATLAB 编译器转换为 C 或 C++代码。

为函数求值使用 feval 比使用 eval 要好些，由于 feval 函数是专门的求值函数，故 feval 函数完成此功能时做了优化。

（2）一系列变量赋值。

给一系列变量命名并赋值的常用方法为用同一变量添加后缀，如 phase1、phase2 和 phase3 等。建议使用细胞数组创建此类型的变量，可使代码可读性好，执行速度快。例如：

```
for k = 1:800
    phase{k} = expression;
end
```

（3）在仅需要计算逻辑表达式的值时，使用逻辑与“&&”和逻辑或“||”要比使用逻辑与“&”和逻辑或“|”高效。

下面的例子中，若 A 为假，则 MATLAB 不会计算 D 的真假值。

```
comp = (A && D)
```

（4）在一个循环体内不能修改计数器的值。

比如，在下面的例子中，虽然在每次循环体中计数器 k 都被赋值为 1，循环体仍只执行了 10 次。

```
for k = 1:10
    disp(sprintf('Pass %d', k))
    k = 1;
end
```

虽然 MATLAB 允许在循环体内使用和计数器同名的变量，但是为了避免混乱和发生难以察觉的错误，最好不要这样使用。

8.2.8 MATLAB 路径

（1）当给出一个名字，如 Hello，MATLAB 会如何解释呢？当成变量还是函数呢？此时，MATLAB 按照下面的顺序来查找这个名字，若查找到后立即停止继续查找并按当前结果解释此名字。顺序如下。

①变量。

②子函数。

③私有函数。

④类的构造函数。

⑤重载函数。

⑥当前路径的 M 文件。

⑦搜索路径上的 M 文件或 MATLAB 内嵌函数。

（2）文件优先级。

若仅给出了文件名，而没有后缀，同时路径上又有不止一个此名字的文件，如仅给出文件名 plot，而此路径上有两个 plot 名的文件 plot.m 和 plot.dll，则 MATLAB 按照下面的顺序来确定此文件。

①MEX 文件

②MDL 文件，即 Simulink 模型

③P 码文件

④M 文件

（3）可通过执行【File】→【Set Path】或【addpath】命令将一个路径添加到搜索路径，这样，此路径下的文件就能被 MATLAB 搜索到。

使用 addpath 函数可将某路径和其全部子路径加入到搜索路径，如：

```
genpath('K:/toolbox/control'))
```

8.2.9 程序控制

（1）break、continue 和 return 的比较。

①break：for 和 while 循环中使用，退出 for 或 while 循环体，在嵌套函数中，退出到外层循环。

②continue：for 和 while 循环中使用，忽略此次循环的其他语句，进入下一个循环。

③return：终止函数运行，控制权交还给调用者。

（2）switch 和 if 的比较。

①可读性：switch 易读，if 难读（相对来说）。

②比较不同长度的字符串：switch 可以，而 if 需要 strcmp 函数支持。

③功能：switch 仅能测试相等性，而 if 可测试相等也可测试不相等。

（3）case 语句与 C 的不同。

在 MATLAB 中，可以 case 表达式可用字符串，而 C 则不行。如：

```
switch(method)
    case 'linear'
        disp('Method is linear')
    case 'cubic'
        disp('Method is cubic')
end
```

在 C 语言中，每个 case 语句后需要加 break 语句来结束下面 case 语句的测试。若不加，即使找到匹配的 case，C 语言也继续测试下面的 case，若下面还有条件合适的 case，C 语言同样执行。在 MATLAB 中 case 语句则不必要，MATLAB 至多会执行一个 case 语句。

在 MATLAB 中不仅没必要使用 break，而且是非法的并生成一个警告。下面的例子中，若结果 result 值为 52，仅第一个 disp 表达式执行，虽然第二个表达式也是符合执行条件的。

```
switch(result)
    case 52
        disp('result is 52')
    case {52, 78}
        disp('result is 52 or 78')
end
```

这些与 C 语言的不同，编程时尤其要注意。

（4）一个 case 语句可以测试多个条件，如：

```
switch(method)
    case {'linear', 'bilinear'}
        disp('Method is linear or bilinear')
    case (<and so on>)
end
```

（5）switch 语句中变量的作用域

由于 MATLAB 仅仅执行至多一个 case 语句，所以某个 case 语句定义的变量在此 switch 结构中的其他 case 语句是不可见的。在 if-else if 语句中同样如此。

下面的例子中若 choice 等于 2，则会报错，错误信息为 x 没有定义。

```
-- SWITCH-CASE --                    -- IF-ELSEIF --
switch choice
    case 1                           if choice == 1
        x = -pi:0.01:pi;             x = -pi:0.01:pi;
    case 2                           elseif choice == 2
        plot(x, sin(x));             plot(x, sin(x));
end
```

（6）try-catch 可嵌套，如下面的例子在发现第一个错误时试图处理此错误：

```
try
    statement1                       % 执行 statement1
catch
    try
        statement2                   % 试图处理错误
    catch
        disp 'Operation failed'      % 未能处理错误
    end
end
```

（7）用 return 强制提前退出函数。如：

```
if <done>
    return
end
```

（8）尽量避免使用循环。

循环语句及循环体经常被认为是 MATLAB 编程的瓶颈问题。改进这样的状况有下面几种方法。

- 尽量用向量化的运算来代替循环操作。在 8.4.2 节将详细介绍。
- 在必须使用多重循环的情况下，如果两个循环执行的次数不同，则建议在循环的外环执行循环次数少的，内环执行循环次数多的。这样也可以显著提高速度。
- 将循环部分编写成等价 MEX 文件。

（9）大型矩阵最好预先定维。

大型矩阵动态地定维是个很费时间的事。建议在定义大矩阵时，首先用 MATLAB 的内嵌函数，如 zeros()或 ones()对之先进行定维，然后再进行赋值处理，这样会显著减少所需的时间。

（10）优先考虑内嵌函数。

矩阵运算应该尽量采用 MATLAB 的内嵌函数，因为内嵌函数是由更底层的编程语言 C 构造的，其执行速度显然快于使用循环的矩阵运算。

（11）采用有效的算法。

在实际应用中，解决同样的数学问题经常有各种各样的算法。例如，求解定积分的数值解法在 MATLAB 中就提供了两个函数：quad()和 quad8()，其中，后一个算法在精度、速度上都明显高于前一个算法。所以，在科学计算领域是存在“多快好省”的途径的。如果一个方法不能满足要求，可以尝试其他方法。

（12）应用 MEX 技术。

虽然采用了很多措施，但执行速度仍然很慢，比如，耗时的循环是不可避免的，这样就应该考虑用其他语言，如 C 或 Fortran 语言。按照 MEX 技术要求的格式编写相应部分的程序，然后通过编译链接，形成在 MATLAB 可以直接调用的动态连接库（DLL）文件，这样可以显著地加快运算速度。

8.2.10 矩阵的操作

由于 MATLAB 的基本数据类型是矩阵，并且 MATLAB 提供了很多针对矩阵的优化函数和算法。正确、灵活地使用矩阵会大大增加程序的可读性。

1．获取矩阵的子矩阵

获取矩阵的子矩阵有 3 种方法：下标法、线性法和逻辑法。

首先介绍下标法，下标法相对比较简单，如下例：

```
A = 6:12;
A =
     6     7     8     9    10    11    12
A([3:2:end])
ans =
     8    10    12
```

线性法是利用了二维矩阵以列优先顺序可以线性展开，通过线性展开后的元素序号来访问元素。A(6)相当于 A(:)结果的第六个元素。如下例：

```
A = [11 14 17;12 15 18; 13 16 19]
A(6)
ans =
    16
A([6;9])
ans =
    16
    19
```

逻辑法则是用一个和原矩阵具有相同尺寸的 0-1 矩阵来索引元素。在某个位置上为 1 表示选取元素，为 0 则不选。得到的结果是一个向量。 如下例：

```
A = 6:10;
A(logical([0 0 1 0 1]))
ans =
     8    10
```

或

```
A=[1 2;3 4];
B=[0 1 0 1]
A(logical(B))
ans =
     3     4
```

2．数组操作和矩阵操作

对矩阵的元素一个个孤立进行的操作称做数组操作；而把矩阵视为一个整体进行的运算则成为矩阵操作。MATLAB 运算符*、/、\和^都是矩阵运算，而相应的数组操作则是.*、./、.\和.^。注意，相对于矩阵操作符前都有一个英语的句点“.”。

如下例：

```
A = [1 0 ;0 1];
B = [0 1 ;1 0];
A*B          %矩阵乘法
ans =
     0     1
     1     0
A.*B         %A 和 B 对应项相乘，即代数中的“点乘”
ans =
     0     0
     0     0
```

3．布朗数组操作

对矩阵的比较运算是数组操作，也就是说，是对每个元素孤立进行的。判断真假就是判断数组的每个元素的真假，所以，其结果就不是一个“真”或者“假”，而是一堆“真假”。这个结果就是布朗数组。

如下例：

```
D = [-0.2 1.0 1.5 3.0 -1.0 4.2 3.14]
D =
   -0.2000    1.0000    1.5000    3.0000   -1.0000    4.2000    3.1400
D>=0
ans =
     0     1     1     1     0     1     1
```

如果想选出 D 中符合条件的元素，而不仅仅得出判断结果，则：

```
D=D(D>0)
D =
    1.0000    1.5000    3.0000    4.2000    3.1400
```

除此之外，MATLAB 运算中会出现 NaN、Inf 和-Inf。对它们的比较参见下例：

```
NaN ~= NaN      %返回 1
NaN == NaN      %返回 0
```

Inf-Inf 和 Inf/Inf 都会产生 NaN：

```
Inf==Inf         %返回 1
Inf>1            %返回 1
```

同时，可以用 isinf 和 isnan 对结果进行判断是否是 Inf 和 NaN。另外，在比较两个矩阵大小时，矩阵必须具有相同的尺寸，否则会报错。

4. 由向量构建矩阵

在 MATLAB 中创建常数矩阵非常简单，经常使用的是：

```
A = ones(5,5)*10
```

其实这个乘法是不必要的：

```
A = 10;
A = A(ones(5,5))
A =
     10    10    10    10    10
     10    10    10    10    10
     10    10    10    10    10
     10    10    10    10    10
     10    10    10    10    10
```

另外的例子如下：

```
v = (1:5)';
n=4;
M = v(:,ones(n,1))
M =

     1     1     1     1
     2     2     2     2
     3     3     3     3
     4     4     4     4
     5     5     5     5
```

事实上，上述过程还有一种更容易理解的实现方法：

```
A = repmat(10,[5 5]);
M = repmat([1:5]', [1,4]);
```

其中，repmat 的含义是把一个矩阵重复平铺，生成较大矩阵。

5. 排序、设置和计数

以下介绍一些可用的基本函数。

- max：最大元素。
- min：最小元素。
- sort：递增排序。
- unique：寻找集合中互异元素（去掉相同元素）。
- diff：差分运算符[X(2) - X(1), X(3) - X(2),⋯,X(n) - X(n-1)]。
- find：查找非零、非 NaN 元素的索引值。
- union：集合并。
- intersect：集合交。
- setdiff：集合差。
- setxor：集合异或。

6. 稀疏矩阵（Sparse Matrix）

在某些情况下，可以使用稀疏矩阵来提高计算的效率。如果构造一个大的中间矩阵，通常矢量化更加容易。在某些情况下，可以充分利用稀疏矩阵结构来矢量化代码，而对于

这个中间矩阵不需要大的存储空间。

8.3　MATLAB 类和面向对象编程

MATLAB 语言中有类数据类型，能创建类的对象，同时也支持面向对象编程语言的重载、继承、封装和聚集等特点。所以，MATLAB 支持面向对象编程。MATLAB 的面向对象编程与 C++等面向对象编程语言有很大相似。

本节讲解 MATLAB 面向对象编程的基础，包括创建类和对象、类方法的重载和继承，最后简要介绍了对象装载保存和优先级等内容。

8.3.1　类和对象

类是面向对象程序设计的核心，实际上它是一种新的数据类型，也是实现抽象类型的工具，因为类是通过抽象数据类型的方法来实现的一种数据类型。类是对某一类对象的抽象；而对象是某一种类的实例，因此，类和对象是密切相关的。没有脱离对象的类，也没有不依赖于类的对象。

类是一种复杂的数据类型，它是将不同类型的数据和与这些数据相关的操作封装在一起的集合体。这有点像结构，唯一不同的就是结构没有定义所说的“数据相关的操作”。“数据相关的操作”就是类中的“方法”。因此，类具有更高的抽象性，类中的数据具有隐藏性，类还具有封装性。

类的结构（也即类的组成）是用来确定一类对象的行为的，而这些行为是通过类的内部数据结构和相关的操作来确定的。这些行为是通过一种操作接口来描述的（也即类的成员函数），使用者只关心接口的功能，即类各个成员函数的功能，而对其具体实现并不感兴趣。而操作接口又被称为这类对象向其他对象所提供的服务。

MATLAB 的类与面向对象编程与 C++或 Java 语言相似，但也有诸多不同，如 MATLAB 中无 C++和 Java 中的方法分发（Method Dispatching）语法，无析构函数，无虚类，无 C++中的域运算符“::”，无 C++中的模板。

8.3.2　创建类和对象

创建一个新的类需要创建一个@classname 目录，并至少提供两个函数 M 文件，第一个是 class.m，该文件用于定义在新类中变量的生成。因此该 M 文件被称为构造器。第二个 M 文件是 display.m，该文件用于在控制窗口中显示新变量。如果没有附加的 M 文件，那么变量类将没有任何用处，但是实际上，仅提供构造器和 display.m 就可以创建一个类。

例 8.56　创建一个多项式类，类名为 polynom，来实现多项式的特殊运算。

首先建立一个类名为名字的目录：@polynom，即类名前加“@”符号，来代表是个类的目录。

多项式用一个行矢量按 x 的指数降序存储多项式各项系数来表示，比如向量[2 0 -4 0 5]

表示多项式 $2x^4-x^2+5$。这样，polynom 对象 p 是一个具有单一字段 p.c 结构，其中 c 为各项系数，并且只有@polynom 目录下的方法才能访问本字段。

为完成多项式的特殊运算，需要实现以下基本方法。

- 构造函数 polynom.m。
- display 方法。
- double 方法，实现 polynom 类型向 double 型的转换。
- char 方法，实现 polynom 类型向 char 型的转换。
- 重载+、-、*、/等运算符来实现多项式的加、减、乘、除。

1．构造函数

下面是 polynom 类的构造函数，文件名为@polynom/polynom.m。

polynom.m

```
function p = polynom(a)
%   POLYNOM Polynomial 类的构造函数
%   p = POLYNOM(v)  由向量 v 创建一个 polynom 对象，按照 x 指数降序存储了各项的系数
if nargin == 0
    p.c = [];
    p = class(p,'polynom');
elseif isa(a,'polynom')       % 检查输入参数是否为 polynom 对象
    p = a;
else
    p.c = a(:).';
    p = class(p,'polynom');
end
```

构造函数先检查输入参数个数，若无则建造一个空多项式，若参数本身为一个 polynom 对象，返回此对象，否则将向量的值作为系数建造一个多项式。建立好此构造函数后，就可以来创建 polynom 对象了，如：

```
p = polynom([1 0 -2 -5])
```

2．转换器方法

转换器方法将 MATLAB 的类对象转换为另外的类的对象，其中常用的是转换为 double 型和 char 型，前者利于生成 MATLAB 矩阵，后者便于输出或打印。

将 polynom 类转换为 double 型，由 M 文件@polynom/double.m 实现，比较简单，仅仅将系数提取出来即可，内容如下：

double.m

```
function c = double(p)
c = p.c;
```

对于上面的 polynom 的对象 p：

```
double(p)
ans =
     1     0    -2    -5
```

转换为 char 型比较复杂，将生成类似于 $2x^4-4x^2+5$ 的字符串，M 文件为

@polynom/char.m，内容如下：

char.m

```
function s = char(p)
if all(p.c == 0)
    s = '0';
else
    d = length(p.c) - 1;
    s = [];
    for a = p.c;
        if a ~= 0;
            if ~isempty(s)
                if a > 0
                    s = [s ' + '];
                else
                    s = [s ' - '];
                    a = -a;
                end
            end
            if a ~= 1 | d == 0
                s = [s num2str(a)];
                if d > 0
                    s = [s '*'];
                end
            end
            if d >= 2
                s = [s 'x^' int2str(d)];
            elseif d == 1
                s = [s 'x'];
            end
        end
        d = d - 1;
    end
end
```

对于上面的对象 polynom 对象 p：

```
char(p)
ans =
    x^3 - 2*x - 5
```

3．display 方法

display 方法用来输出结果到屏幕，下面是 display 的 M 文件@polynom/display.m，此 M 文件利用到 char 函数生成的字符串，然后显示出来，内容如下：

display.m

```
function display(p)
disp(' ');
disp([inputname(1),' = '])
disp(' ');
disp(['     ' char(p)])
```

```
disp(' ');
```

用构造函数构造上面的 polynom 的对象 p 时，若后面不加分号，则 MATLAB 输出如下：

```
p =
     x^3 - 2*x - 5
```

8.3.3 重载

与其他面向对象语言一样，MATLAB 提供了函数重载和运算符重载的支持。在 MATLAB 类的目录下创建 M 文件可实现类的函数重载。如在类目录下创建并编辑 M 文件 plot.m，因为 MATLAB 在调用函数时，先在本目录下搜索，所以此函数优先于 MATLAB 的 plot 函数执行，进而实现了针对本类的 plot 函数。

对于上面的例 8.56，这一节将继续完善，重载 plus（+）、minus（-）和 mtimes（*）算术运算符后以实现多项式的加、减、乘、除等运算。

1．重载算术运算符

1）重载“+”运算符

P 和 *Q* 为 polynom 的两个对象，若想用 *P+Q* 来实现两个多项式的加法，则必须重载 plus 运算符。

实现函数为 M 文件@polynom/plus.m，内容如下：

plus.m

```
function r = plus(p,q)
% POLYNOM/PLUS   实现 polynoms 对象 p 和 q 的加法  p + q

p = polynom(p);
q = polynom(q);
k = length(q.c) - length(p.c);
r = polynom([zeros(1,k) p.c] + [zeros(1,-k) q.c]);
```

此函数首先确认参数均为 polynom 对象，使得表达式

```
p+1
```

也能正确运算。将“短”的多项式，即最高项次数低的多项式与“长”的多项式相比，然后将所缺项的系数设为 0，使得两个多项式“对齐”后，对应系数相加。最后利用构造函数得到结果。

2）重载“－”运算符

同样，若简单调用 *P*－*Q* 来实现两个多项式的减法，也必须重载 minus 运算符。

实现函数为 M 文件@polynom/minus.m，内容如下：

minus.m

```
function r = minus(p,q)

% POLYNOM/MINUS 实现 polynoms 对象 q 和 p 的减法  p - q
p = polynom(p);
q = polynom(q);
```

```
k = length(q.c) - length(p.c);
r = polynom([zeros(1,k) p.c] - [zeros(1,-k) q.c]);
```

实现方法与 plus 一样，只是最后一步是相应系数相减。

3）重载“*”运算符

若简单调用 $P*Q$ 来实现两个多项式的乘法，则需重载 mtimes 运算符。

实现函数为 M 文件@polynom/mtimes.m，内容如下：

mtimes.m

```
function r = mtimes(p,q)

% POLYNOM/MTIMES   实现 polynoms 对象 q，p 的乘法 p * q..
p = polynom(p);
q = polynom(q);
r = polynom(conv(p.c,q.c));
```

其中 conv 函数来实现多项式的相乘。

2. 重载函数

MATLAB 本身有专门操作多项式的函数，重载它们可以为 polynom 类所用。

1）重载 root 方法

roots.m

```
function r = roots(p)
% POLYNOM/ROOTS.   ROOTS(p)求 p 的根
r = roots(p.c);
```

对于上面的对象 p：

```
root(p)
ans =
        -1.0000 + 2.6458i
        -1.0000 - 2.6458i
```

2）重载 polyval 方法

polyval 方法计算在给定点集处多项式的值。M 文件为@polynom/polyval.m，内容如下：

```
function y = polyval(p,x)
% POLYNOM/POLYVAL   polyval(p,x)计算得出 p 在 x 点处的值

y = 0;
for a = p.c
    y = y.*x + a;
end
```

3）重载 plot 方法

此 plot 方法调用了 root 方法和 polyval 方法，文件名为@polynom/plot.m，内容如下：

```
function plot(p)
% POLYNOM/PLOT   plot(p) 绘出多项式 p.

r = max(abs(roots(p)));
x = (-1.1:0.01:1.1)*r;
y = polyval(p,x);
```

```
plot(x,y);
title(char(p))
grid on
```

4）重载 diff 方法

方法@polynom/diff.m 为多项式求导，方法为次数减 1，然后系数乘以原次数，内容如下：

diff.m

```
function q = diff(p)
% POLYNOM/DIFF   diff(p) 给多项式 p 求导
c = p.c;
d = length(c) - 1;   % 次数
q = polynom(p.c(1:d).*(d:-1:1));
```

函数调用：

```
methods('classname')
```

或命令行调用：

```
methods classname
```

会显示此类的所有方法。对于上面的 polynom 类：

```
methods polynom
Methods for class polynom:
char    display   minus     plot  polynom    roots
diff    double    mtimes    plus  polyval    subsref
```

下面的代码先对多项式对象 x 和 p 进行操作，然后调用 plot 输出最后的多项式：

```
x = polynom([1 0]);
p = polynom([1 0 -2 -5]);
plot(diff(p*p + 10*p + 20*x) - 20)
```

生成如图 8-6 所示的曲线。

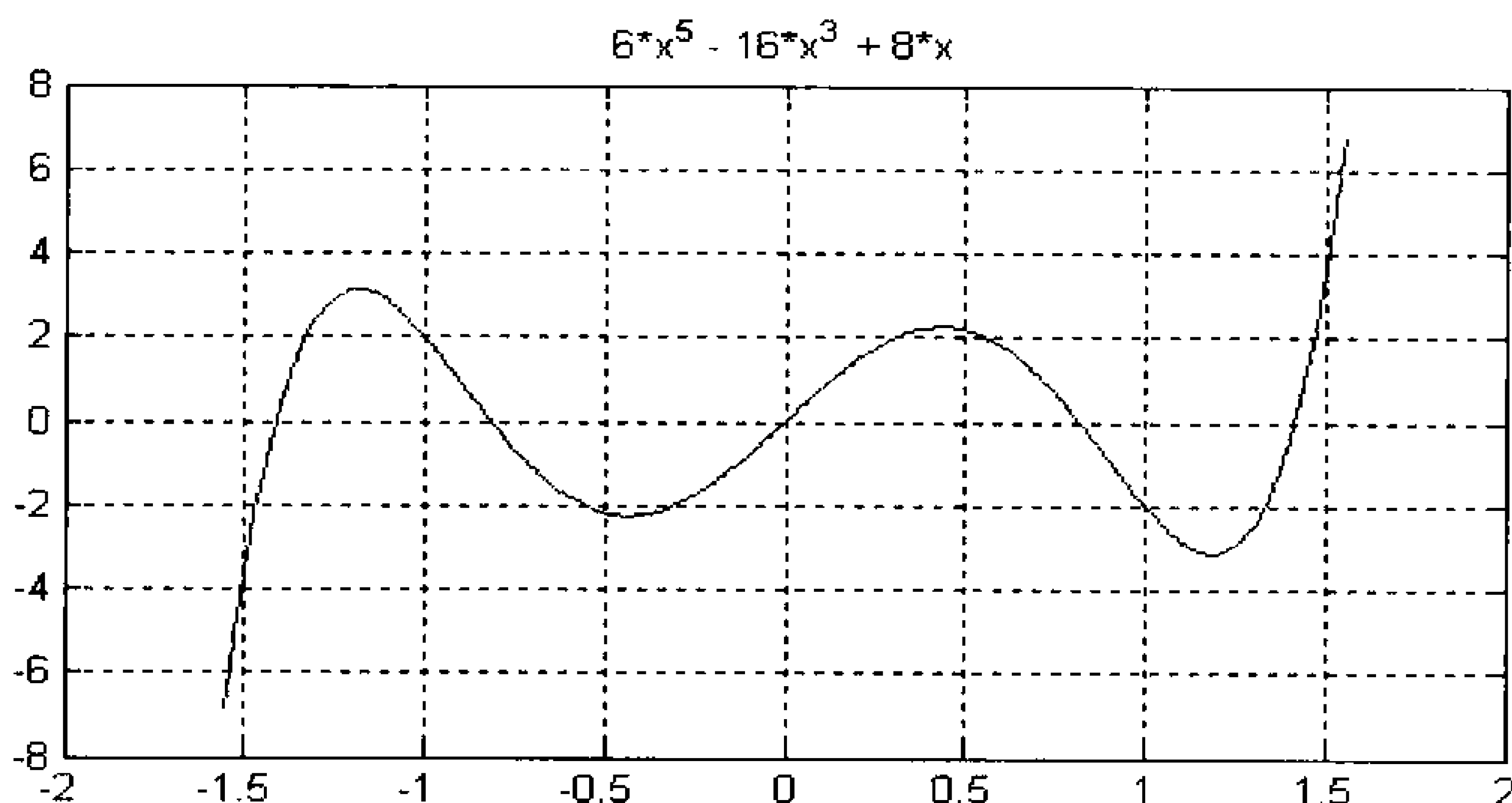


图 8-6　多项式 p 和 x 运算后的多项式曲线

8.3.4　继承

与 C 语言类似，MATLAB 的类的属性和行为可从另外一个类继承而来。被继承的类称为父类或基类（Base Class），而继承类称为子类。继承时，子对象拥有父对象的所有属性

（Fields），而且可以调用父对象的所有方法。父对象也可以访问子对象中从父对象继承而来的属性，而不能访问子对象独有的属性。

继承是面向对象编程的重要特征，通过简单继承不需额外编写代码，子类就可以完成基类中的功能，从而使代码重用变得简单。

继承有两种类型，一种是单继承，其中子类只有一个父类；另一种是多继承，子类由多个基类继承而来，此时子类拥有多个父类的综合特征。

1．单继承

若一个类继承一个基类的属性并添加自己新的属性，属于单继承。在单继承中，子类从基类继承来的属性和基类中的原属性一样。

基类的方法可以对子类的对象操作，但子类的方法却不能对基类的对象操作。访问基类的对象，必须用基类的方法。

子类的构造函数中必须调用基类的构造函数创建继承来的属性，同时，子类的 class 函数语法此时有所区别，包括基类的特点和子类的特点。

一般用 class 来创建单继承的语法如下：

```
derivedObj = class(derivedObj, 'derivedClass', baseObj);
```

单继承可跨越多层继承，若某子类的基类同样是另一个类的子类，那么此子类同样也继承了其祖父类的所有特征。

2．多继承

在多继承中，子类对象从所有的基类中继承属性，所以子类对象拥有所有基类的属性和自己的属性。多继承也可以有多个继承层次。

多继承多用于 3 个参数的 class 函数实现：

```
obj = class(structure,'classname',baseclass1,baseclass2,...)
```

后面可以根据需要添加任意多个参数。

由于子类从多个基类中继承了方法，若两个或多个基类中有相同名字的方法，此时 MATLAB 采用构造函数参数表中第一个出现的基类的该方法，而后面的该方法忽略，也不能被访问。

3．示例

例 8.57 创建一个资产类 Asset，Asset 类表示任何有“钱财”属性的东西。并由 Asset 派生出股票（Stock）、奖金（Bond）和存款（Savings）3 个子类。

此例说明单继承的使用方法和相关技巧。设计基类 Aseet 时，要考虑到所有这些子类的所有属性。Asset 的子类，如 Stock 类，包含了基类的属性，并具有自己独有的属性，子类和基类直接是“一种（Kinds of）”的关系。其中子类 Stock 类、Bond 类和 Saving 类与基类 Asset 类的继承关系如图 8-7 所示。

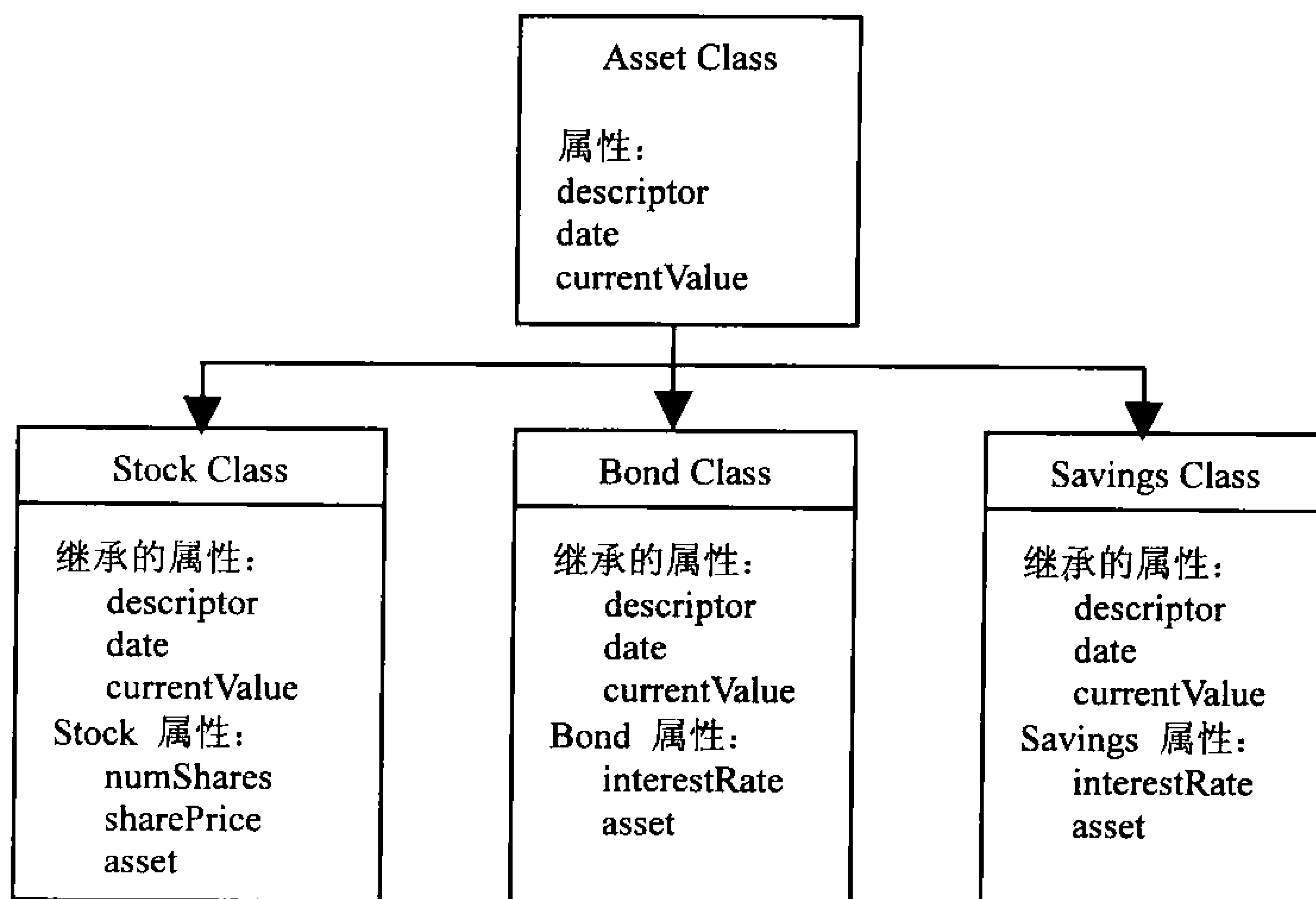


图 8-7　基类 Asset 和子类直接的继承关系

下面简要介绍基类 Asset 类和一个子类 Stock 类的设计，其他子类设计与 Stock 子类的设计类似。

1）Asset 类的设计

Asset 类为所有子类提供了存储和访问的方法，由于基类抽象了子类的公有属性，所以基类不需要十分具体的方法，而只需要提供构造函数、获取数据和设置数据的 get 和 set 方法以及用来显示的 display 方法等基本方法。

Asset 类有以下 4 个属性。

- Desciptor：比如股票名字，账户号码等。
- date：对象创建日期。
- type：Asset 的类型，比如 savings、bond 和 stock。
- currentValue：Asset 的当前金额。

这些属性是其子对象共有的，所以在基类中定义避免了在子类中的重复定义。在子类的数目增加时尤为有用。

Asset 类的构造函数如下：

```
function a = asset(varargin)
% ASSET Asset 类的构造函数，调用方式为： a = asset(descriptor, type, currentValue)
switch nargin
case 0
% 若无输入参数，创建默认对象
    a.descriptor = 'none';
    a.date = date;
    a.type = 'none';
    a.currentValue = 0;
    a = class(a,'asset');
case 1
    if (isa(varargin{1},'asset'))
        a = varargin{1};
    else
```

```
            error('Wrong argument type')
        end
    case 3
        a.descriptor = varargin{1};
        a.date = date;
        a.type = varargin{2};
        a.currentValue = varargin{3};
        a = class(a,'asset');
    otherwise
        error('Wrong number of input arguments')
    end
```

构造函数用 switch 语句来测试输入参数的个数，若无参数，则创建默认对象，若有一个参数，则返回，若有 3 个参数，根据参数值创建新的对象。

Asset 类的 get 方法来获取具体的对象属性的值，内容如下：

```
function val = get(a, propName)
% GET  获取对象某属性的值
switch propName
case 'Descriptor'
    val = a.descriptor;
case 'Date'
    val = a.date;
case 'CurrentValue'
    val = a.currentValue;
otherwise
    error([propName,' Is not a valid asset property'])
end
```

此方法接收一个对象名和属性名，根据属性名来获取指定属性的值。

与 get 方法相对应的是 set 方法，set 方法将 Asset 类的属性设置为给定的数据值。

```
function a = set(a,varargin)
% SET  设置 Asset 类属性，并返回修改后的对象
propertyArgIn = varargin;
while length(propertyArgIn) >= 2,
    prop = propertyArgIn{1};
    val = propertyArgIn{2};
    propertyArgIn = propertyArgIn(3:end);
    switch prop
    case 'Descriptor'
        a.descriptor = val;
    case 'Date'
        a.date = val;
    case 'CurrentValue'
        a.currentValue = val;
    otherwise
        error('Asset properties: Descriptor, Date, CurrentValue')
    end
end
```

display 方法为子类所设计，作用是显示调用对象存储的数据。此方法对数据的格式进行了格式化，以便对调用者的数据保持格式一致。

```
function display(a)
% DISPLAY(a)  显示 Asset 对象
stg = sprintf('Descriptor: %s\nDate: %s\nType: %s\nCurrent Value:%9.2f',...
    a.descriptor,a.date,a.type,a.currentValue);
disp(stg)
```

2）子类 Stock 的设计

Stock 类设计成 Asset 类的子类，表示股票投资，包括本身两个属性和从 Asset 类继承来的 3 个属性。Stock 独有的属性有 NumberShares 和 SharePrice，分别表示特定股票的数目和每份股票的价格。继承来的属性有 Descriptor、Date 和 CurrentValue，分别为股票名字、创建日期和当前金额。

Stock 同样也需要构造函数，get 和 set 方法以及 display 方法。不过其具体实现可由基类 Asset 已有的方法来帮助实现。

Stock 类的构造函数需要 3 个输入参数，股票名、股票份额和股票价格。Stock 的构造函数必须创建一个 Asset 对象作为父对象，所以必须调用 Asset 的构造函数。Stock 类的构造函数是在临时工作空间中创建的 Asset 对象，以属性形式（.asset）存储，Stock 对象继承了 Asset 类的属性，但是 Asset 对象并不返回到基本工作空间。

Stock 的构造函数如下：

```
function s = stock(varargin)
% STOCK Stock 类的构造函数，调用方式为：s = stock(descriptor, numShares, sharePrice)
switch nargin
case 0
    s.numShares = 0;
    s.sharePrice = 0;
    a = asset('none',0);
    s = class(s,'stock',a);
case 1
    if (isa(varargin{1},'stock'))
        s = varargin{1};
    else
        error('Input argument is not a stock object')
    end
case 3
    s.numShares = varargin{2};
    s.sharePrice = varargin{3};
    a = asset(varargin{1},'stock',varargin{2} * varargin{3});
    s = class(s,'stock',a);
otherwise
    error('Wrong number of input arguments')
end
```

可以这样调用构造函数创建 Stock 对象：

```
XYZStock = stock('XYZ',100,25);
```

同样，Stock 类的 get 方法和 set 方法提供了获取/设置股票名、股票数和股票价格的途径。具体实现分别如下：

```
function val = get(s,propName)
% GET Get stock property from the specified object
% and return the value. Property names are: NumberShares
```

```
    % SharePrice, Descriptor, Date, CurrentValue
    switch propName
    case 'NumberShares'
        val = s.numShares;
    case 'SharePrice'
        val = s.sharePrice;
    case 'Descriptor'
        val = get(s.asset,'Descriptor'); % call asset get method
    case 'Date'
        val = get(s.asset,'Date');
    case 'CurrentValue'
        val = get(s.asset,'CurrentValue');
    otherwise
        error([propName ,'Is not a valid stock property'])
    end
```

set 方法：

```
    function s = set(s,varargin)
    % SET Set stock properties to the specified values
    % and return the updated object
    propertyArgIn = varargin;
    while length(propertyArgIn) >= 2,
        prop = propertyArgIn{1};
        val = propertyArgIn{2};
        propertyArgIn = propertyArgIn(3:end);
        switch prop
        case 'NumberShares'
            s.numShares = val;
        case 'SharePrice'
            s.sharePrice = val;
        case 'Descriptor'
            s.asset = set(s.asset,'Descriptor',val);
        otherwise
            error('Invalid property')
        end
    end
    s.asset = set(s.asset,'CurrentValue',...
        s.numShares * s.sharePrice,'Date',date);
```

在用新值创建了一个新的对象时，需要对旧的对象进行覆盖。因为 MATLAB 仅支持值传递，而不支持引用方式传递参数。比如 Stock 对象 s：

```
    s = stock('XYZ',100,25);
```

下面的表达式更新了股票价格：

```
    s = set(s,'SharePrice',36);
```

若不更新，原对象的值并不会更新，因为 MATLAB 仅仅对原对象的一个复制进行了操作，而没有修改原本的对象，原对象值保持不变。

Stock 的 display 方法如下：

```
    function display(s)
    % DISPLAY(s) Display a stock object
    display(s.asset)
```

```
stg = sprintf('Number of shares: %g\nShare price: %3.2f\n',...
    s.numShares,s.sharePrice);
disp(stg)
```

如在控制窗口中输入以下命令来调用 display 方法：

```
XYZStock = stock('XYZ',100,25)
```

得出以下结果：

```
Descriptor: XYZ
Date: 17-Nov-1998
Type: stock
Current Value:   2500.00
Number of shares: 100
Share price: 25.00
```

可见，调用 Stock 类的 display 方法时，会先调用 Asset 类的 display 方法显示 Asset 的属性。所以若不具体实现 Stock 的 display 方法，也可以进行显示，由于是 Asset 的 display 方法，只仅仅显示从 Asset 继承来的属性。

8.3.5 聚集

除了标准的继承之外，MATLAB 的对象支持聚集（Aggregation），即一个对象可以包含另外一个对象作为其他的一个属性。比如，有理数 rational 的对象可能包含两个多项式 polynom 对象，一个作为分子，一个作为分母。MATLAB 只能将最外层的对象作为参数传递，内层的对象属性被忽略。

聚集实现了一个类作为另一个类的容器，容器中的类是容器的“一部分”，下面以一个实例介绍聚集的使用方法。

例 8.58 实现一个简单的 Portfolio 类，Portfolio 来描述资产，包含某人的各种资产，如 stocks、bonds 和 savings 等。除构造函数外，还能显示各资产的信息。

Portfolio 的构造函数，需要对以下属性进行初始化。

- Name：客户名称。
- indAssets：Aseet 子对象数组。
- totalValue：所有资产的总额。
- accountNumber：账号。

其构造函数如下：

```
function p = portfolio(name,varargin)
% PORTFOLIO 创建一个 portfolio 对象，包含客户名和 Aassets 列表
switch nargin
case 0
    p.name = 'none';
    p.totalValue = 0;
    p.indAssets = {};
    p.accountNumber = '';
    p = class(p,'portfolio');
case 1
    if isa(name,'portfolio')
        p = name;
```

```
        else
            disp([inputname(1) ' is not a portfolio object'])
            return
        end
    otherwise
        p.name = name;
        p.totalValue = 0;
        for k = 1:length(varargin)
            p.indAssets(k) = {varargin{k}};
            assetValue = get(p.indAssets{k},'CurrentValue');
            p.totalValue = p.totalValue + assetValue;
        end
        p.accountNumber = '';
        p = class(p,'portfolio');
    end
```

由此可见，Portfolio 的构造函数和 Asset 构造函数类似。

Portfolio 的 display 方法列出所有对象的内容，然后列出客户名和资产总额：

```
function display(p)
% DISPLAY  显示 portfolio 对象
for k = 1:length(p.indAssets)
    display(p.indAssets{k})
end
stg = sprintf('\nAssets for Client: %s\nTotal Value: %9.2f\n',...
p.name,p.totalValue);
disp(stg)
```

假如已经像 Stock 类一样实现了 Asset 其他的子类，就可以用一个 Portfolio 对象列出资产信息，如给定下面的资产信息：

```
XYZStock = stock('XYZ', 200, 12);
SaveAccount = savings('Acc # 1234', 2000, 3.2);
Bonds = bond('U.S. Treasury', 1600, 12);
```

创建一个 Portfolio 对象：

```
p = portfolio('Gilbert Bates',XYZStock,SaveAccount,Bonds)
```

Portfolio 的 display 方法列出了 Portfolio 对象的所有内容：

```
Descriptor: XYZ
Date: 24-Nov-1998
Current Value:     2400.00
Type: stock
Number of shares: 200
Share price: 12.00
Descriptor: Acc # 1234
Date: 24-Nov-1998
Current Value:     2000.00
Type: savings
Interest Rate: 3.2%
Descriptor: U.S. Treasury
Date: 24-Nov-1998
Current Value:     1600.00
Type: bond
```

```
Interest Rate: 12%
Assets for Client: Gilbert Bates
Total Value:    6000.00
```

8.3.6 对象保存与装载

1．保存和装载对象

像操作一般变量一样，MATLAB 可用 save 和 load 命令来保存用户定义的对象到 MAT 文件和从 MAT 文件将用户定义的对象载入到工作空间。载入对象时，MATLAB 调用对象的类的构造函数在工作空间中注册这个对象。被调用的构造函数必须可以无参数调用而且返回一个默认对象。

2．在保存或装载中修改对象

当用命令来进行保存或载入对象操作时，MATLAB 在类的路径中查找类的方法 saveobj 和 loadobj。重载这些方法可以在载入或保存对象之前修改对象。

比如，可以定义一个 saveobj 方法来同时保存一些相关的数据，或者写一个 loadobj 方法使得对象载入到 MATLAB 空间前更新成一个新的版本。

8.3.7 对象优先级

对象优先级解决在给定的情况下如何选择不同的操作符和函数的版本。优先级使得可以控制包含不同类的对象的表达式的操作行为。比如，考虑下面的表达式：

```
objectA + objectB
```

通常 MATLAB 默认对象有相同的优先级，调用最左边的对象定义的操作符。即上面的表达式中的“+”将解释为对象 A 中所定义的。但是，有如下两种意外情况。

- 用户定义的类优先于 MATLAB 内嵌（Built-in）的类。
- 用户定义的类之间可以定义相互优先级关系。

例 8.59 polynomial 类中的加法，polynom 类定义了多项式对象之间的 plus 方法。对于 polynom 的对象 p：

```
p = polynom([1 0 -2 -5])
p =
    x^3-2*x-5
```

则表达式“1 + p”的结果如下：

```
1 + p
ans =
     x^3-2*x-4
```

调用了 polynom 类的 plus 方法（将 double 型的 1 转化成了 polynom 对象，然后与 p 相加），用户定义的 polynom 类的优先级高于 MATLAB 本身的 double 类。

在类的构造函数中使用 feriorto 和 superiorto 函数可以对用户定义的类定义相对优先级关系。inferiorto 函数语法如下：

```
inferiorto('class1','class2',...)
```

排在前面的类的优先级低于后面的类。参数列表中可以有多个类。

类似地，superiorto 函数的语法如下：

```
superiorto('class1','class2',...)
```

排在前面的类的优先级高于后面的类。

如果 objectA 的优先级高于 objectB，那么下面的表达式

```
objectA + objectB
```

会调用@classA/plus.m。而若 objectB 的优先级高于 objectA，那么 MATLAB 会调用@classB/plus.m。

8.4 改善运行效率与内存利用

MATLAB 是一种解释语言，比起 C、Fortran 等编译语言的执行效率要低。MATLAB 编程语法和 C 语言在很大程度上相似，使得初学者以 C 语言编程的习惯来编写 MATLAB 程序，而只有充分认识和利用 MATLAB 本身的特点，才能编写高效的程序。

本节首先介绍了MATLAB程序的性能分析方法，然后分别以实例介绍了提高MATLAB运行效率和 MATLAB 程序中如何有效利用内存。

8.4.1 程序性能分析

利用 M 文件评述器（Profiler）和秒表函数（Stopwatch Timer Functions）可以测出程序的性能，并找出需要提高性能的代码段。M 文件评述器适用于测出程序各段执行的相对时间，从而找出制约程序效率的瓶颈，秒表程序则可以测出某程序段执行需要的绝对时间。

1. M 文件评述器

提高程序效率的有效方法是找出制约 M 文件效率的瓶颈所在。尽量优化瓶颈部分可以大幅度地提高整个程序的执行效率。MATLAB 提供了 M 文件评述器，可以统计出程序中每个函数执行所需的时间。通过运行时间的对比可找出需要优化的代码。

在 MATLAB 的控制窗口输入 profile viewer 可以打开评述器窗口，或者利用菜单和工具栏。

2. 秒表函数

若需要测试出程序运行所需时间，或对不同的运行方式所需时间进行对比，则可利用秒表函数 tic 和 toc。tic 函数启动定时器，第一个紧跟它的 toc 函数终止定时器并报告此时定时器的流逝时间。其语法如下：

```
tic
   -- 所要测试的程序段 --
toc
```

比较短的程序执行较快，所以利用 tic 和 toc 得到的数据误差偏大，此时可以重复地运行此程序，计算总的运行时间，然后计算每次运行的平均值。例如：

```
tic
   for k = 1:100
      -- 运行程序--
   end
toc
```

虽然利用 cputime 函数也可测试程序性能，但是建议仅仅使用 tic 和 toc 函数。

8.4.2 提高运行效率

下面给出几种利用 MATLAB 特点而优化代码提高效率的方法。

1．计算使用多线程

在多处理器系统或多核系统上运行 MATLAB 程序时，可以编写多线程程序来利用多线程计算能力。

MATLAB 计算默认是单线程的，允许多线程计算需要如下设置：执行【File】→【Preferences】命令，在对话框中执行【General】→【Multithreading】→【Enable multithreaded computation】命令。然后还可以设置线程的个数。

2．矢量化操作

MATLAB 是个矩阵语言，内部数据运算是基于矢量和矩阵的。算法采用矢量化（Vectorization）利用了 MATLAB 的这个特点，从而可以加速 M 文件执行。矢量化是将 for 和 while 循环转化成等价的矢量或矩阵运算。

例 8.60 计算步长为 0.01，从 0~10 的 1001 个数值的 sin 值。

下面是一般的做法：

sine.m

```
i = 0;
for t = 0:.01:10
    i = i + 1;
    y(i) = sin(t);
end
```

将上面的代码等价向量化，代码如下：

sine2.m

```
t = 0:.01:10;
y = sin(t);
```

矢量化后执行得要比前者快很多。将上面代码分别写成脚本文件并结合 tic 和 toc 函数，可以测试出二者的速度对比。笔者自己测试结果为 0.029658s 和 0.000275s（不同机器测试会有不同），矢量化后加速了一百多倍。

例 8.61 用 repmat 函数扩展矩阵 A。

repmat 函数是矢量化的一个例子，有 3 个参数，矩阵 A、行数 M 和列数 N。repmat 创建了一个输出矩阵，此矩阵为元素都为 A 的“$M \times N$”的矩阵。如：

repmatsam.m

```
A = [1 2 3; 4 5 6];
B = repmat(A,2,3);
B =
     1     2     3     1     2     3     1     2     3
```

```
4    5    6    4    5    6    4    5    6
1    2    3    1    2    3    1    2    3
4    5    6    4    5    6    4    5    6
```

repmat 使用了矢量化创建输出矩阵的索引。repmat 函数内部执行的步骤是：

```
    function B = repmat(A, M, N)
% 第一步，获取矩阵的行和列大小：
    [m,n] = size(A);
% 第二步，生成行矢量和列矢量：
    mind = (1:m)';
    nind = (1:n)';
% 第三步，根据上面的矢量创建输出矩阵索引：
    mind = mind(:,ones(1, M));
    nind = nind(:,ones(1, N));
% 第四步，根据索引值创建输出矩阵：
    B = A(mind,nind);
```

对于这个函数的实现，并不像一般想象的那样用 for 循环或者 while 来赋值。

例 8.62　利用 repmat 提取一幅 RGB 图像中的蓝色部分。

repmatsam2.m

```
A=imread('input.bmp');
H = size(A);
BB=repmat(255, H(1)*H(2), H(3));
C=(A(:,:,1)==0 & A(:,:,2)==0 & A(:,:,3)==255);
BB(C,1) = 0;
BB(C,2) = 0;
B = reshape(BB, H(1), H(2), H(3));
imwrite(B,'output.bmp');
```

注意 repmat 和 reshape 的用法。解答的关键是对 reshape 命令的灵活使用，把三维矩阵转换为二维矩阵，从而使得求出满足条件的下标矩阵 ***C*** 可以对原矩阵 ***A*** 进行引用。否则，这类问题一定摆脱不了循环语句。

例 8.63　把矩阵 ***A*** 作 0 扩展（变身：1 个变 4 个，原来真身维持不变，其他假身为 0）。比如：

```
A =
     1    2    3    4
    -1   -2   -3   -4
     0    4    2    9
```

希望得到：

```
B=
   1    2    3    4    0    0    0    0
  -1   -2   -3   -4    0    0    0    0
   0    4    2    9    0    0    0    0
   0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0
```

则做法如下：

repmatsam3.m

```
B = repmat(zeros(size(A)), 2, 2);
B(1:size(A,1), 1:size(A,2)) = A;
```

不推荐的方法如下：

repmatsam4.m

```
[row, col] = size(A);
for i = 1:2*row
    for j = 1:2*col
        if i >=1 && i <= row && j >= 1 && j <= col
            B(i, j) = A(i, j);
        else
            B(i, j) = 0;
    end
end
```

也许是受制于 C 语言的影响，初学者往往把 MATLAB 当做 C 语言来使用，加上 MATLAB 变量不需要定义即可以直接使用，因此导致大量的循环+变量判断语句。掌握“:”运算符是向 MATLAB 矢量化编码进军的必经之路。

例 8.64 有个 80*80 的矩阵，去掉全为 0 的行或者列。

假设有矩阵 ***a***：

```
a =
    0    0    0   -3    0
    1    0    0    0    0
    0    2    0    0    0
    0    0    0    0    0
    0    0    0    0    0
```

解法如下：

```
a(:,~sum(abs(a),1))=[ ]
a =
    0    0   -3
    1    0    0
    0    2    0
    0    0    0
    0    0    0
a(~sum(abs(a),2),:) = [ ]
a =
    0    0   -3
    1    0    0
    0    2    0
```

此例又是一个摆脱循环思想的典型例子。除了“:”运算符的妙用之外，逻辑法对矩阵的下标索引也相当重要。此外，线性代数的知识也在这里得到体现：

$|a|+|b|+|c|=0$，当且仅当 $a=b=c=0$ 时。

3．预分配矩阵空间

for 和 while 循环中若每步增加或修改矩阵的维数，会大大影响执行效率和内存使用率。重复地修改维数需要 MATLAB 浪费多余时间查找更大的连续可用的内存空间，然后把当前矩

阵移动到这些空白区。所以预先分配矩阵可能使用的最大空间，可以大大提高代码执行效率。

下面的代码创建一个向量变量 x，然后在一个 for 循环中增加 x 的维数，而不是在开始时预先分配内存空间。

```
x = 0;
for k = 2:1000
    x(k) = x(k-1) + 5;
end
```

修改第一行，初始化 x 时，先预分配一个 1×1000 的内存块，并初始化为 0。此时，当 x 逐步赋值时，就不必要像上面那样重复地重新分配内存空间并移动数据了。

```
x = zeros(1, 1000);
for k = 2:1000
    x(k) = x(k-1) + 5;
end
```

预分配矩阵空间的函数有 zeros 和 cell。Zeros（***M*，*N***）创建一个 ***M***×***N*** 的矩阵，并初始化为 0 矩阵。cell 用来创建细胞矩阵（Cell Matrix）。

4．给数组预定义维数

与 C 语言不同，MATLAB 允许用户可以先使用一个变量，而不必预先定义其维数。但是每当赋值的元素的下标超出了现有矩阵的维数时，MATLAB 就要为该矩阵扩充一次。有时候会要查找更大的连续内存空间并移动数据，这样势必影响程序的执行效率，而且多次移动数据后会造成不必要的内存碎片，从而影响内存使用率。

```
i = 0;
y=size(1,1000);
for t = 0:.01:10
     i = i + 1;
     y(i) = sin(t);
end
```

用 tic 和 toc 函数对上面的代码进行测试，执行时间从原来的 0.029658s 减少到预定义维数后的 0.003870s，效率大大提高了。

5．将循环转化为 C-MEX 体

虽然 for 循环和 while 要尽量矢量化，但是有些情况时必须使用 for 或 while 循环，此时为了提高程序执行效率，可以将循环部分的代码转化为 C-MEX 文件。C-MEX 是可执行文件，在 Windows 环境下，它是扩展名为.dll 的动态链接库，可以在 MATLAB 环境下直接执行。这样，循环体部分不必每次执行前再解释（Interpret），故比转化前执行要快很多。编写 C-MEX 文件要复杂得多，好在 MATLAB 提高了将 M 文件转化为 C-MEX 文件的工具。这部分读者请参考第 10 章 MATLAB 外部接口部分。

6．变量赋值

为获得更高性能，为变量赋值时需特别注意下面两个问题。

1）改变变量的数据类型或维数

改变已有变量的数据类型或矩阵维数会使 MATLAB 降低效率。若需要保存变量为另外不同的类型，建议创建一个新的变量。下面的代码操作使变量 X 的类型从 double 改成了 char，这个做法会降低 MATLAB 的效率。

```
X = 23;
other code  --
X = 'A';                    % X 的类型 从 double 改成了 char
other code  --
```

推荐的做法是：

```
X = 23;
other code  --
XX = 'A';                   %创建一个新的变量 XX，再赋值
other code  --
```

2）为 real 型和复数赋值

把一个已有变量赋值为复数也会降低 MATLAB 的效率。类似地，不能将一个复数值的变量赋值为实数值。

7．选择正确的逻辑操作符

当需要执行 AND 或 OR 逻辑操作时，有两种类型可以选择，AND 可以选择&或&&，OR 可以选择 | 或 ||。

```
x = A&B
y = A&&B
```

在计算 x 的值时，分别计算 A 和 B 的值，然后再判断 x 的值。而在计算 y 的值时，若发现 A 为假，则不再计算 B 的值，因为此时已经可以得知 y 的值为假。| 和 || 的区别类似。由于这个区别，省去了一些不必要的操作，故在 if 和 while 表达式中，使用&&进行逻辑 AND 操作，和 || 进行逻辑 OR 操作更有效率。比如：

```
if (nargin >= 3) && (ischar(varargin{3}))
```

8．不要重载内嵌（Built-in）函数

重载 MATLAB 的内嵌函数会降低 MATLAB 的效率。比如重载 plus 函数，使得 plus 进行不同的整型数据计算，就会妨碍 MATLAB 对内嵌函数 plus 所做的优化。所以使用此重载函数的程序会降低效率。

9．尽量使用函数文件

在 MATLAB 中，函数文件的效率一般比脚本文件的效率要高。因为函数文件有自己的工作空间，执行一次后就保存了程序运行必需的变量，并且函数编译成了伪代码，所以下次调用时就提高了效率。所以在编程时，尽量使用函数文件而非脚本文件。

10．尽量使用 load 和 save 函数

load 和 save 函数和低层次的 MATLAB 文件 I/O 函数，比如 fread 和 fwrite 相比，前者要快很多。因为 load 和 save 经过了 MATLAB 的优化，运行更快，并且内存碎片少。

8.4.3 改善内存利用

MATLAB 独特的数据结构以及其他特点，使得 MATLAB 在进行大型运算时会占用大量的内存。合理地使用内存和提高内存使用率对内存进行有效管理，可以加快程序的运行速度，减少对资源的占用。

1．数组的内存分配

MATLAB 的基本数据类型是矩阵，掌握数组的内存分配机制对有效使用内存特别重要。

当为一个变量赋值任意类型的数据，MATLAB 为之分配一个连续的内存块并存储数据。同时，MATLAB 在一个单独的小的内存块保存了此变量的有关信息，比如，数据类型和维数。这个小的内存块称为数据头（Header）。被赋值的变量实质上是数据的一个“指针”，并没有包含此数据。

对于结构体和细胞数组，MATLAB 不仅为整个数据创建数据头，同时也为结构体的每个域或细胞数组的每个元素创建数据头。所以存储结构体和细胞数组型数据所需的内存不仅取决于数据的多少，而且和数据的组织形式有关。

比如，结构体 S1 有 R、G、 B 三个域，每个域的大小都是 100×50 的矩阵。S1 有一个整体的数据头，同时 R、G、B 3 个域各有一个数据头，故存储 S1 额外需要 4 个数据头：

```
S1.R(1:100,1:50)
S1.G(1:100,1:50)
S1.B(1:100,1:50)
```

另外的一个结构体数组 S2，大小为 100×50，每个元素都有 3 个域 R、G、B。除了整体数据头之外，结构体的 5000 个元素的每个域也需要一个数据头，这样，存储 S2 额外需要 1+100×50×3＝15001 个数据头：

```
S2(1:100,1:50).R
S2(1:100,1:50).G
S2(1:100,1:50).B
```

虽然 S1 和 S2 包含同样多的数据，S1 明显地使用了更少的内存。不仅如此，使用 S1 的数据结构也会加快操作速度。

2．数据类型与内存使用

存储不同类型的 MATLAB 数据结构需要不同的内存。故了解 MATLAB 的数据存储方式有利于减少内存使用。

MATLAB 分别使用 1、2、4 和 8 字节存储 8-bit、16-bit、32-bit 和 64-bit 的有符号和无符号整型。单精度浮点型数据占用 4 字节，而双精度占用 8 字节。为了节省内存，在保证不溢出的情况下，建议使用最小类型的数据结构。MATLAB 分别存储实部和虚部来存储复数。另外，最好把多数是 0 值的矩阵存储成稀疏矩阵。稀疏矩阵占用更少的内存而且便于数据操作。

比较两个 1000×1000 的矩阵，*X* 有 2/3 的元素为 0，*Y* 是 *X* 转化成的稀疏矩阵。下面是占用内存的对比，稀疏矩阵几乎减少了一半的内存占用。

```
  Name      Size                Bytes   Class
  X         1000x1000         8000000   double array
  Y         1000x1000         4004000   double array (sparse)
```

3．内存管理函数和命令

下面的函数可以管理 MATLAB 的内存。

- whos：显示基本工作空间变量占用的内存。
- pack：把变量存盘，需要时导入，减少了内存碎片。
- clear：清除内存中的变量。定期清除内存中不需要的变量，可以有效地节省内存。

- save：将所需变量存盘。操作大量数据时，先将所需变量存盘，然后用 clear 命令清除内存中的变量，也可以有效地节省内存。
- load：导入由 save 命令保存的数据文件。
- quit：退出 MATLAB 并释放 MATLAB 占用的所有内存。此命令在 UNIX 系统中尤为有用，因为 UNIX 直到程序退出时才会释放程序所占的内存。

4．节约内存的策略

节约内存有以下几方面策略。

（1）避免创建占内存大的中间变量，当临时变量不再需要时从内存中清除。

（2）在使用能够确定大小的数组时，预先指定大小并分配内存，避免频繁增加维数而产生内存碎片。即避免下面代码形式：

```
for t = 0:1000
    y(t) = sin(t);
end
```

而最好提前指定 y 的大小为 1000。

（3）尽量优先为大型矩阵分配内存。

比如，下面的表达式会占用大约 32.4 MB 的 RAM：

```
a = rand(1e6,1);
b = rand(1e6,1);
clear
c = rand(2.1e6,1);
```

但是如果先为大型矩阵 **c** 分配内存，则仅占用了 16.4 MB 的 RAM：

```
c = rand(2.1e6,1);
clear
a = rand(1e6,1);
b = rand(1e6,1);
```

（4）给变量赋值为空矩阵“[]”以释放内存，或者使用 clear 函数清理内存中的变量。

（5）当程序需要生成大量变量数据时，可以考虑定期将变量写到磁盘，然后清理这些变量。当需要这些变量时，重新从磁盘加载。

8.5　M 文件调试与剖析

在开发 M 文件程序时，尤其是团队合作大型项目时，不可避免地会出现错误，也就是所谓的 bug。MATLAB 提供了几种有助于调试 M 文件的方法和函数。

本节介绍了编程中的一般错误处理方法和直接调试法，最后结合实例介绍了 MATLAB 调试器的使用和 M 文件性能剖析报告的使用。

8.5.1　错误处理（Error Handling）

简单地检测错误发生的方法是用 try-catch 结构。如计算矩阵乘法：

```
function matrixMultiply(A, B)
try
```

```
    X = A * B
catch
    disp '** Error multiplying A * B'
end
```

另外，try-catch 结构可以嵌套，try-catch 结构在 8.1.3 节 MATLAB 控制流中有详细介绍。

若仅仅是报告出错信息，error 语句比较有用，如：

```
if n < 1
    error('n must be 1 or greater.')
end
```

当 error 语句报错后，需要查找出出错原因，以便正确处理出错信息。用 lasterror 可返回出错的详细信息。error 的一般使用方法如下：

```
function MyFunction(A, B)
try
    -一些操作-
catch
    err = lasterror;
    fprintf('   信息: %s\n\n', err.message)
    fprintf('   标识: %s\n\n', err.identifier)
    errst = err.stack;
    fprintf('   出错在\n')
    fprintf('         行  %d\n', errst.line)
    fprintf('         函数  %s\n', errst.name)
    fprintf('         文件  %s\n', errst.file)
end
```

lasterror 返回一个结构体，包含出错文本、出错标识、出错文件名、出错行以及出错的函数。

例 8.65　利用 lasterror 显示矩阵乘法错误的类型。

```
function matrixMultiply(A, B)
try
    A * B
catch
    err = lasterror;
    if(strfind(err.message, 'Inner matrix dimensions'))
        disp('** Wrong dimensions for matrix multiply')
    else
        if(strfind(err.message, ...
                   'not defined for values of class'))
            disp('** Both arguments must be double matrices')
        end
    end
end
```

若两个矩阵的维数不匹配，会得到下面的出错信息：

```
A = [1  2  3; 6  7  2; 0  1  5];
B = [9  5  6; 0  4  9];
matrixMultiply(A, B)
** Wrong dimensions for matrix multiply
```

若有一个矩阵为细胞矩阵，则会得到以下出错信息：

```
C = {9 5 6; 0 4 9};
matrixMultiply(A, C)
** Both arguments must be double matrices
```

8.5.2 直接调试法

编写 M 文件时一般会出现以下两种错误。

（1）语法错误（Syntax Error）。语法错误包括变量名、函数名不合法，标点符号使用不正确或错、漏。遇到此类错误，MATLAB 会终止程序的运行，然后提供错误信息，包括错误类型和 M 文件中出错代码的行号。利用这些信息，错误通常就能很方便地进行定位。

（2）运行错误（Runtime Error）。运行结果和预期效果不相符。原因可以是多方面的，包括对算法理解不正确、误用指令和程序流控制不合理等。由于这种错误 MATLAB 不会给出错误信息，很难发现，而需要跟踪调试，找出问题。

例 8.66 求两数的最大公约数和最小公倍数。

```
A1= input('输入两个非零数 第一个:');
A2= input('输入两个非零数 第二个:');
a=max(A1,A2);
b=min(A1,A2);
while(b~=0)                                    %利用代数中的辗转相除法
   r=rem(a,b);
   a=b;
   b=r;
end
disp(a);
disp(A1*A2/a);
```

其实 MATLAB 中得到两个整数相除，即 A/B 的余数有专门函数 rem(A,B)，而有 C++ 编程习惯的程序员会误用 A%B，这样最后将得不到正确结果。

另一方面，运行时错误在通常情况下很难被发现，即使 MATLAB 会将它们标记出来。当运行时一个错误被发现后，MATLAB 将控制权返还给控制窗口和 MATLAB 工作空间。用户也无法访问错误发生的函数工作空间。因此无法通过查询这个函数的工作空间来试图隔离出现的问题。

调试函数 M 文件有多种方法，对于简单问题，用下述的一种或者几种方法来解决是非常直观的。

（1）将函数中被选定的行的分号去掉，这样运算的中间结果就可以在控制窗口汇总显示出来。

（2）在 M 文件中选定的位置置入 keyboard 命令，以便将临时控制权交给键盘，这样做以后，函数工作区就可以进行查询，并且可以根据需要改变变量的值。在键盘提示符下输入 return 命令就可以恢复函数执行，也就是说，在 k>>下输入 return 就可以了。

（3）通过在 M 文件开头的函数定义语句之前插入%，把函数 M 文件变成脚本 M 文件。在以脚本 M 文件执行的时候，其工作空间就是 MATLAB 基本工作空间，这样在出现错误的时候就可以查询这个工作空间了。

（4）在适当的位置利用命令显示变量值，例如，直接以变量名作为一行、利用 disp 命令等。

（5）利用 echo on 和 echo off 显示执行的指令行，判断程序流是否正确。

8.5.3 调试器的使用

当 M 文件很大时，M 文件是可以递归调用或者高度嵌套的，也就是说它调用其他 M 文件函数，而这些函数又调用其他函数，等等。使用 MATLAB 的编译器的【Debug】和【Breakpoints】菜单中的图形化调试是非常方便的。这些函数在控制窗口的等价函数也存在，但是用起来要麻烦得多。如果用户坚持使用这些函数而不是图形化的调试器，可在控制窗口输入 helpwin debug 以得到 debug 的帮助信息。

下面是 MATLAB 的一些调试命令。

- dbstop：设置断点。
- dbclear：移除断点。
- dbcont：恢复执行。
- dbdown：改变当前工作空间，进入下一层。
- dbmex：允许 MEX 文件调试。
- dbstack：列出调用关系（who called whom）。
- dbstatus：列出所有断点。
- dbstep：按行执行。
- dbtype：列出 M 文件内容，包括行序号。
- dbup：改变当前工作空间，进入上一层。
- dbquit：退出调试模式。

MATLAB 的图形化的调试工具允许用户在已设置的断点和出现 MATLAB 报警和错误的地方，以及得到了 NaN 和 Inf 的表达式的地方中断程序的运行。当程序运行到断点的时候，MATLAB 在断点所在的行进行运算并返回结果之前停止执行。一旦停止了，控制窗口中就显示出键盘提示符 k>>，使得用户可以查询函数工作空间，改变工作空间的变量的值，等等。还可以根据需要跟踪调试。另外，编辑器/调试器窗口还显示出运行中断的行，并提供了进入当前这个调试的 M 文件调用的其他函数的工作空间的方法。当用户准备好了从这个断点向下运行的时候，编辑器/调试器窗口还提供了用来单步执行、运行到下一个断点停止、运行直到到达指针位置、结束调试等菜单条。

描述图形化调试工具的用法是一件很困难的事情，这是因为它们的图形化特征和调试过程在每一个调试过程中都是不同的。但是这些调试工具还是比较直观并且简单易用的，一旦用户掌握了其中的基本步骤，MATLAB 的图形化调试工具就能在生成运行良好的 MATLAB 代码的过程中体现出其强大和高效的特性。

与 Visual C++的调试环境类似，MATLAB 支持在程序中设置断点，然后可以按步执行，执行每一步后，都可在 MATLAB 控制窗口查看此时工作空间的变量情况。由于 MATLAB 的脚本文件和 MATLAB 共享基本工作空间，而函数文件有单独工作空间，所以查看变量信息时要看当前工作空间是否是所要的工作空间。对于上面的求最大公约数的例子，改造成

函数文件后如下：

Fgcd.m

```
function [m,n]=gcd(x,y)
a=max(x,y);
b=min(x,y);
while(b~=0)
    r=rem(a,b);
    a=b;
    b=r;
end
m=a
n=x*y/a
```

调试时，如在 MATLAB 的控制窗口输入 *a* 想查看 *a* 的值时，会得到下面错误信息：

```
??? Undefined function or variable 'a'.
```

因为工作空间不同的原因，此时需要 dbdown 和 dbup 命令来改变当前工作空间。查看变量信息的另外一个比较方便的方法是利用 MATLAB 的变量值提示（Detatips)。MATLAB 有特色的地方是在调试状态时，当鼠标停留在某个变量上时，MATLAB 会自动给出此变量的信息，包括类型和值，使得查看变量更为方便。

使用 Evaluate Selection 可查看变量或表达式的值。在 M 文件编辑器中选中某变量或表达式，然后用鼠标右键选择 Evaluate Selection 或单击工具栏中相应图标可以得出变量此时的值，结果显示在 MATLAB 控制窗口中。

在程序调试状态时，可以修改变量的值，然后根据此新值进行运算，查看修改值后的运算结果。按步执行时，可以修改变量值，然后继续执行。在命令行、工作空间浏览器或者数组管理器中都可以对变量赋新值。

MATLAB 另外的一个特色是可以设置条件断点和错误断点。条件断点就是在程序满足指定的条件时停止执行。错误断点是程序出现错误时自动进入调试状态，并且 MATLAB 会停留在程序出错的地方。添加一般断点的方法是在 M 文件编辑器中执行【debug】→【Set/Clear Breakpoint】命令，或单击所要添加断点的行的行号后面的小短横线。添加条件断点的方法是先单击要添加断点的行，然后执行【debug】→【Set/Modify Breakpoint】命令，然后在弹出的对话框中输入程序要停止的条件即可。一般断点的行头以红色圆圈标注，条件断点的行头以黄色圆圈标注。

例 8.67　判断循环次数，避免死循环。

while 循环条件若设置不当，可能会造成死循环。对于上面的求最大公约数和最小公倍数的例子，若用错误的运算符%，而 a%b=a，即“余数”永远不会等于 0，while 语句就会陷入死循环。可以加入循环次数判断，若大于某个比较大的值，就以为陷入了死循环，则停止程序执行，进入调试状态。

looptime.m

```
A1= input('ÊäÈëÁ½¸ö·ÇÁãÊý µÚÒ»¸ö:');
A2= input('ÊäÈëÁ½¸ö·ÇÁãÊý µÚ¶þ¸ö:');
a=max(A1,A2);
```

```
b=min(A1,A2);

i=0;
while(b~=0&&i<1000)
    r=a%b;                %加入对 i 值的判断，防止死循环
    a=b;
    b=r;
    i=i+1
end
```

对于本例，由于正常地求余数，余数会小于除数，所以加入这个判断语句可以更好地防止错误的发生：

```
i=0;
while(b~=0)
    r=a%b;               % 在此加入对“余数”的判断的条件断点
    a=b;
    b=r;
end
```

8.5.4 M 文件性能剖析

调试器只负责 M 文件中语法错误和运行错误的定位，而性能剖析指令 profile 将给出程序各环节的耗时分析报告。MATLAB 剖析指令的分析报告特别详细，它将帮助用户寻找影响程序运行速度的“瓶颈”所在，以便改进。

例 8.68 剖析二阶系统阶跃响应的 M 文件，文件名为 huitu.m。

huitu.m

```
t=[0:0.1:18]';
for x=0.2:0.2:0.8
    b=sqrt([1-x^2]);
    z=atan(b/x);
    y1=-t*x; y2=t*b+z;
    y=1-exp(y1).*sin(y2)/b;
    plot(t,y), hold on
end
xlabel('t (秒)'),ylabel('y')
title('二阶系统阶跃响应')
text(3.3,0.9,'{\xi}=0.8')
text(4.3,1.4,'{\xi}=0.2')
```

多运行几次程序，计算平均值：

profilesam.m

```
profile on                          % 启动剖析器
for k=1:100;                        % 运行 100 次，使统计数据受随机因素影响较小
  huitu;
  end
profile report t                    % 导出分析报告
```

导出报告如图 8-8 所示。

Profile Summary

Generated 28-May-2007 17:34:28 using cpu time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
tutu	100	4.260 s	1.327 s	
xlabel	200	1.479 s	1.392 s	
hold	400	0.550 s	0.340 s	
newplot	400	0.507 s	0.151 s	
gca	1100	0.328 s	0.203 s	
axescheck	1000	0.217 s	0.217 s	
title	200	0.203 s	0.116 s	
gcf	1100	0.200 s	0.200 s	
ylabel	200	0.193 s	0.103 s	

图 8-8　huitu.m 文件的剖析报告

从图 8-8 中可以看出，报告内容十分详细，包括函数调用次数、函数总耗时、函数单独耗时以及总耗时和单独耗时的对比图形。对一个程序进行性能剖析产生类似上述性能报告后，就可以查看哪些函数占用时间较多，哪些函数是整个程序的瓶颈。也就为提高程序性能提供了参考。

8.6　定时器规划程序执行

与 C 语言中的定时器（Timer）类似，MATLAB 也有定时器对象，可用来规划 MATLAB 命令的执行。使用 MATLAB 定时器对象可以更方便地开发结构清晰、运行流畅的程序。

本节介绍创建定时器对象，启动定时器和设置定时器触发事件。定时器触发指的是定时器对象消逝了预先规定的时间，然后 MATLAB 执行预先设置的函数或命令。

8.6.1　MATLAB 定时器

使用 MATLAB 定时器需要以下步骤。

（1）创建定时器对象。

（2）设置定时器，包括定时器的触发事件和其他属性。

（3）启动定时器对象。

（4）删除定时器对象。

例 8.69　定时器的简单例子。

mytimer.m

```
t = timer('TimerFcn', 'stat=false; disp(''定时器触发!'')', 'StartDelay',5);
start(t)                              % 启动定时器
stat=true;
while(stat==true)
   disp('定时器未触发')
```

```
    pause(1)
end
```

本例创建了一个定时器对象，并设置定时器的两个属性 TimerFcn 和 StartDelay。TimerFcn 设置了定时器的回调函数（Callback Function），回调函数可以是 MATLAB 命令也可以是 M 文件。

此例中，定时器的回调函数设置 MATLAB 工作空间中的 stat 变量并执行 disp 命令。StartDelay 属性设置了定时器的触发时间为 5 秒。创建了定时器对象后，使用 start 函数启动定时器对象。

执行 mytimer.m 脚本文件后，在控制窗口输出以下结果：

```
定时器未触发
定时器未触发
定时器未触发
定时器未触发
定时器触发!
```

使用完定时器后，从内存中删除定时器：

```
delete(t)
```

用 timerfind 可以查找出内存中的定时器对象，如：

```
t_array = timerfind
```

返回下面结果：

```
Timer Object Array
    Index:  ExecutionMode:  Period:  TimerFcn:          Name:
    1       singleShot      1        1x32 char array    timer-1
    2       singleShot      1        1x26 char array    timer-2
    3       singleShot      1        1x26 char array    timer-3
```

若定时器的 ObjectVisibility 属性设为“off”，是不可见的，用 timerfind 函数查找不到，此时要用 timefindall 才能查找出内存中不可见的定时器对象。

8.6.2 创建定时器

用 timer 命令可创建定时器对象，并且同时设置定时器属性，语法如下：

```
T = timer('属性名 1', 属性值 1, '属性名 2',  属性值 2,...)
```

另外，在创建定时器后，可用 set 命令设置定时器属性，语法如下：

```
set(t,'TimerFcn','disp("Hello World!")','StartDelay',5)
```

此命令和下面的命令效果一样：

```
t = timer('TimerFcn', 'disp("Hello World!")','StartDelay',5);
```

创建了一个定时器后，MATLAB 为其命名为-1 以区分不同的定时器。第 i 个定时器命名为-i。在创建了两个定时器，然后用 delete 命令从内存中删除后，再创建定时器，这时 MATLAB 仍为其命名-3。若想重新编号，需使用 clear classes 命令。

8.6.3 读取和设置属性

定时器对象支持很多属性，这些属性反映了定时器当前的状态以及控制信息。下面介绍对定时器的属性进行读取和设置方法。

1．读取定时器属性

读取定时器的某个属性，可以使用 get 函数或使用 ObjectName.Property 的方法来读取。

下面代码用 set 函数来读取 ExecutionMode 属性：

```
t = timer;
tmode = get(t,'ExecutionMode')
tmode =
        singleShot
```

下面代码用小句点“.”来设置 ExecutionMode 属性：

```
tmode = t.ExecutionMode
tmode =
        singleShot
```

用 get(t)可以列出定时器对象的所有属性值。如获取例 8.69 中定时器 t 的属性：

```
get(t)
        AveragePeriod:   NaN
            BusyMode:   'drop'
             ErrorFcn:   ''
        ExecutionMode:   'singleShot'
          InstantPeriod:   NaN
                  Name:   'timer-5'     %为节省篇幅，略去后面 12 个属性值 —— 笔者注
```

2．设置定时器属性

与读取定时器属性方法相对应，使用 set 函数或 ObjectName.Property 可以设置定时器属性。也可以在创建定时器的同时设置定时器属性。

例 8.70　用两种方法来设置定时器属性。

（1）使用 set 函数设置属性：

```
set(t,'ExecutionMode','fixedRate','BusyMode','drop','Period',1);
```

（2）用 ObjectName.Property 方式设置 TimerFcn 属性：

```
t.TimerFcn = 'disp(''Processing...'')'
```

用 set 函数可查看可设置的属性（有些属性是只读的），只用定时器作为参数。set 函数返回所有可以用 set 函数来设置的属性。如对于例 8.69 中的定时器 t：

```
set(t)
    BusyMode: [ {drop} | queue | error ]
    ErrorFcn: string -or- function handle -or- cell array
    ExecutionMode: [ {singleShot} | fixedSpacing | fixedDelay | fixedRate ]
    Name
    ObjectVisibility: [ {on} | off ]
    Period
    StartDelay
    StartFcn: string -or- function handle -or- cell array
    StopFcn: string -or- function handle -or- cell array
    Tag
    TasksToExecute
    TimerFcn: string -or- function handle -or- cell array
    UserData
```

8.6.4 启动和停止定时器

启动定时器有两种方式，一种是定时器启动后按秒计时，一种是按特定的时间来计时，比如一个小时、一天。前者可直接用 start 函数来启动，如：

```
t = timer('TimerFcn','disp(''Hello World!'')','StartDelay', 5);
start(t)
```

而后者需用 startat 函数来启动，如：

```
t2=timer('TimerFcn','disp(''It has been an hour now.'')');
startat(t2,now+1/24);                %一小时后触发定时器
```

停止定时器可直接用 stop 函数，如：

```
stop(t)
get(t,'Running')       % 此时获取定时器的运行状态
ans =
      off
```

另外，在启动定时器后，可用 wait 函数来等待定时器的触发，此时 MATLAB 控制窗口被冻结，即不能输入命令，只有等定时器触发后才可使用。

8.6.5 创建和执行回调函数

回调函数除了可直接用 MATLAB 的命令外，还可用 M 文件，包括脚本文件和函数文件。创建回调函数时，前两个参数必须是定时器的句柄和事件结构体，事件结构体包括两个域： Type 和 Data。Type 是标识回调事件类型的字符串，值可以为如表 8-1 所示中的值；Data 为回调函数调用时的时间。

表 8-1 Type 的可取值与含义

Type 值	含 义
StartFcn	启动函数，在定时器启动时回调
StopFcn	停止函数，在定时器停时回调
TimerFcn	触发函数，当定时时间到时回调
ErrorFcn	错误函数，在出错时回调

例 8.71 在回调函数中显示此函数被触发的类型（Type 域的值）和回调函数被调用的时间。

my_callback_fcn.m

```
function my_callback_fcn(obj, event, string_arg)
txt1 = ' event occurred at ';
txt2 = string_arg;
event_type = event.Type;
event_time = datestr(event.Data.time);
msg = [event_type txt1 event_time];
disp(msg)      % 显示回调函数触发方式、触发时间
disp(txt2)     %  显示输入参数 string_arg
```

另外，此函数还有一个字符串参数，用在了 disp 函数中。

在设置 event 的回调函数属性时，可用不同的回调函数类型。回调函数可以为文本字符串、细胞数组或是函数句柄。函数的类型取决于函数定义时的输入参数个数。如表 8-2 所示列出了不同的函数定义与对应的调用方法。

表 8-2　不同的函数类型以及相对应的调用方法

回调函数的语法	设置回调函数属性的语法
function myfile	set(h, 'StartFcn', 'myfile')
function myfile(obj, event)	set(h, 'StartFcn', @myfile)
function myfile(obj, event, arg1, arg2)	set(h, 'StartFcn', {'myfile', 5, 6})
function myfile(obj, event, arg1, arg2)	set(h, 'StartFcn', {@myfile, 5, 6})

例 8.72　演示设置回调函数属性的多种方法。

setdemo.m

```
% 创建定时器对象
t = timer('StartDelay', 4,'Period', 4,'TasksToExecute', 2, 'ExecutionMode','fixedRate');
% 设置 StartFcn 属性。方式为细胞矩阵
t.StartFcn = {'my_callback_fcn', 'My start message'};
% 设置 StopFcn 属性。方式为函数句柄
t.StopFcn = { @my_callback_fcn, 'My stop message'};
% 设置 TimerFcn 属性。方式为 MATLAB 命令字符串
t.TimerFcn = 'disp(''Hello World!'')';
% 启动 timer
start(t)
```

然后输出结果如下：

```
StartFcn event occurred at 10-Mar-2007 17:16:59
Start message
Hello World!
Hello World!
StopFcn event occurred at 10-Mar-2007 17:16:59
Stop message
```

最后删除 timer：

```
delete(t);
```

8.6.6　定时器执行模式

定时器对象支持多种执行模式，执行模式决定了回调函数（TimerFcn）的执行方式。设置定时器的执行模式可用 set 函数设置定时器的 ExecutionMode 属性实现。

1．执行一次回调函数

设置 ExecutionMode 属性为“singleShot”，则回调函数只执行一次。同时“singleShot”是默认模式，在没有设置 ExecutionMode 属性时，MATLAB 认为是 singleShot。在这种模式下，当定时器启动，过了 StartDelay 属性所规定的时间后，将回调函数 TimerFcn 添加到 MATLAB 执行队列。等函数执行完毕后，定时器随即停止工作。

如图 8-9 所示中标识“Queue log”的阴影部分表示从定时器将回调函数加入到 MATLAB

执行队列到回调函数得到执行之间的不确定时间。此段时间长度取决于此时 MATLAB 后台执行的进程。

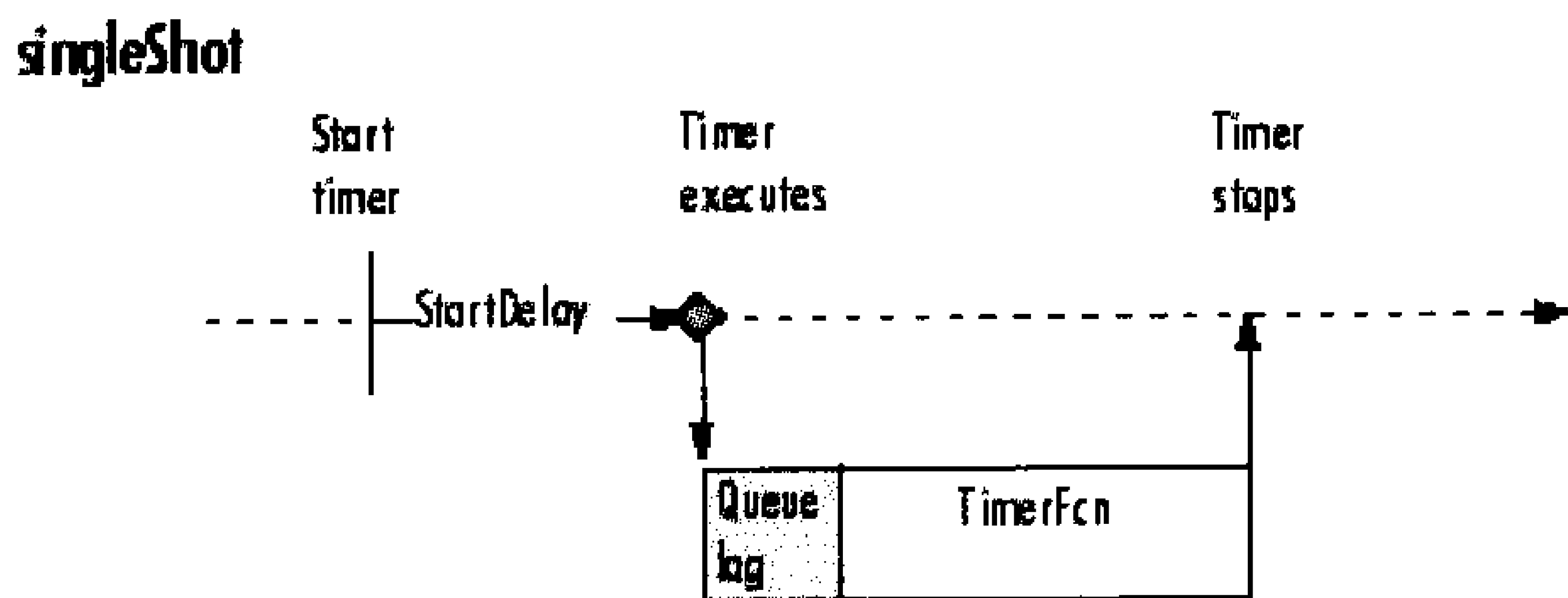


图 8-9　singleShot 执行模式下的回调函数执行部分

2．执行多次回调函数

定时器支持 3 种多次执行的模式：fixedRate、fixedDelay 和 fixedSpacing。

在下面的情况下，这 3 种执行模式是一样的。

- TasksToExecute 属性定义了执行回调函数（TimerFcn）的总次数。
- Period 属性定义了回调函数多次执行之间的间隔。
- BusyMode 属性定义了当回调函数的前次执行未完时，此次执行的排队方式。

这 3 个模式的区别仅仅在于多次函数执行之间的时间间隔的起始值何时开始计算，如表 8-3 所示说明了这些区别。

表 8-3　3 种执行模式的区别

执行类型	时间间隔计算方法
fixedRate	在函数加入到 MATLAB 执行队列后立即计算
fixedDelay	在回调函数实际执行时开始计算
fixedSpacing	在回调函数执行完毕后开始计算

如图 8-10 所示说明了这 3 种模式的区别。多次执行之间的时间间隔（Period 属性）是一样的，唯一的不同是时间间隔的计算方式不同。

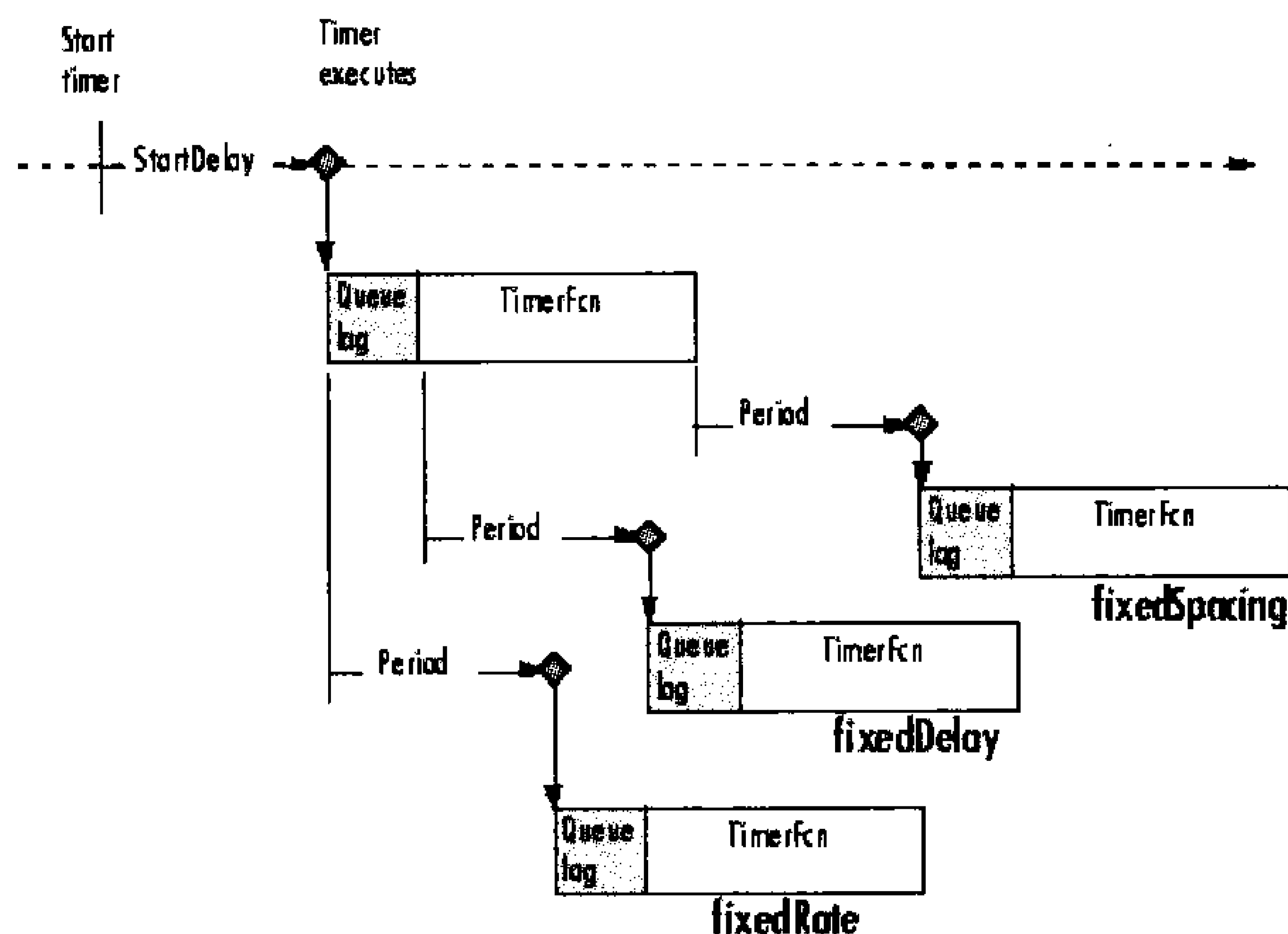


图 8-10　3 种执行模式的区别

由于回调函数需要排队，当回调函数需要执行而上次执行还没有完毕时，排队就出现了冲突。设置 BusyMode 属性可定义出现冲突时不同的处理方法：若设置为“drop”则 MATLAB 放弃此次函数的执行；若设置为“queue”则在队列中等待；若设置为“error”则视为错误，定时器停止，若定义了一个出错误回调函数的话还会调用错误回调函数（ErrorFcn）。

第9章　高级图形用户界面设计

随着面向对象技术的兴起，图形用户界面（Graphical User Interfaces，GUI）设计变得更加流行。许多程序设计软件纷纷推出的图形用户界面设计功能，大大减轻了程序设计者的负担，加快了设计者的程序设计工作。图形用户界面（GUI）是用户与计算机程序之间的交互方式，它是包含图形对象，如窗口、图标、菜单和文本以及工具栏的用户界面。用户以某种方式选择或激活图形对象而引起动作或发生变化。通过图形界面用户可以非常直观、轻松地与计算机交互，且用户不必了解应用程序是如何执行各条命令的，只要掌握图形界面的各个组件的使用方法即可。

MATLAB 也为用户设计图形界面提供了一个高效、方便的集成环境。在 MATLAB 中，基本的图形对象主要包括坐标轴对象(Axes)、控件对象(Uicontrol)、下拉菜单对象(Uimenu）和内容式菜单对象（Uicontextmenu)。用户可以通过这些对象设计出界面友好、功能强大、操作简单的图形用户界面。

本章将介绍如何使用 GUIDE 和图形编程方法创建图形用户界面、MATLAB 的图形用户界面开发环境以及怎样创建一个 GUIs 程序，最后介绍了几个实例。

本章主要内容：

- 入门
- 图形用户界面设计工具
- 对话框
- 界面菜单
- 用户控件
- 图形用户界面编程
- 图形用户界面设计的原则和一般步骤
- 图形用户界面设计实例

9.1　入门

前面章节我们已经提到句柄图形对象，每一个图形对象都有唯一一个句柄（Handle）和一组定义图形对象外观的属性（Properties)。用户可以使用这些句柄图形对象属性来控制图形的行为和外观。对“句柄图形”的理解是设计和实现 GUI 的前提条件。图形对象不仅包括 uimenu 和 uicontrol 对象，而且还包括图形、坐标轴、文本以及它们的子对象。

1．什么是 GUI

GUI 是一个包含了使用户能够执行交互式任务的设备或者组件图形显示。在执行这些任务的时候，用户可以不必在 MATLAB 命令行输入脚本或者命令，用户也可不必知道任务到底是如何执行的详细情况。

GUI 组件主要有 menus、toolbars、push buttons、radiobuttons、list boxes 和 sliders 等。在 MATLAB R2007 中，一个 GUI 可能是以列表形式显示数据的，也可能是以绘图的形式或者是以一组相关联组件的形式显示数据的。如图 9-1 所示为一个简单的 GUI 程序界面。

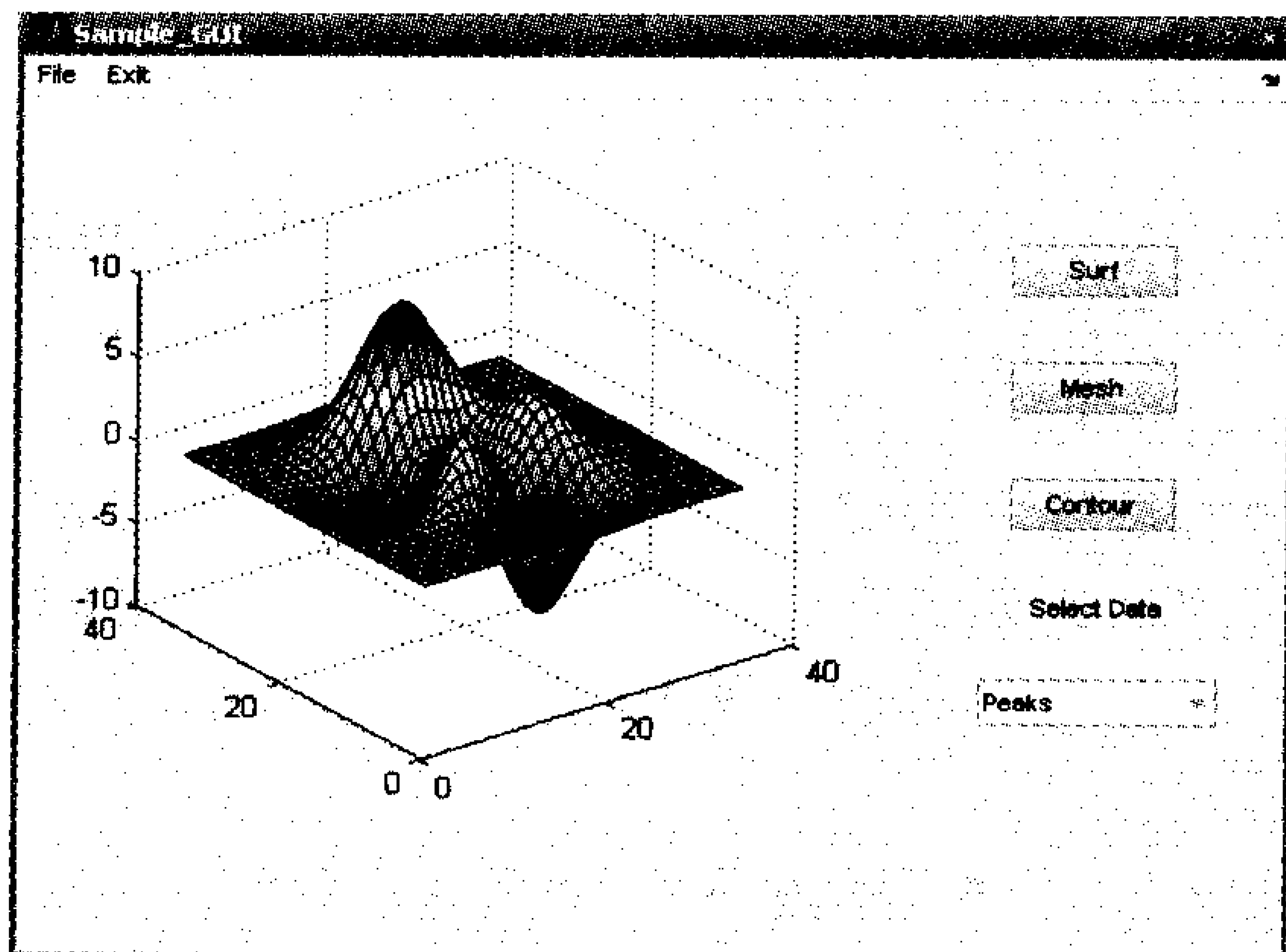


图 9-1 简单的 GUI 程序界面

2．GUI 如何工作

每一个组件，包括 GUI 本身，都与一个或多个用户编写好的程序作为回调函数相关联。每一个回调函数的执行都是由用户的一个行为触发（如单击鼠标），选择菜单项或者是指针移过某个组件等。用户作为 GUI 的创建者编写出这些回调函数。

GUI 编程一般把它作为事件驱动型编程，如图 9-2 所示案例中的事件就是单击按钮。在事情驱动型编程中，回调函数的执行是受事件外部软件控制而不同步的。在 MATLAB 的 GUIs 所有例子中，这些事件常常把它当做用户与 GUI 交互的形式。

3．如何创建 MATLAB 的 GUIs

在 MATLAB R2007 中，给用户提供了使用·GUIDE 的方法或图形用户编程方法来创建自己的 GUIs。我们将在接下来的章节分别介绍如何使用 GUIDE 和编程的方法来创建 GUIs。

在 MATLAB 中启动 GUIDE 有两种方式：一种是命令行方式，另一种是使用 GUI 设计工具。

第一种：在 MATLAB 的命令窗口输入 GUIDE 命令将会得到图 9-2；

第二种：在 MATLAB 主界面执行【File】→【New】→【GUI】命令同样也会出现图 9-2。

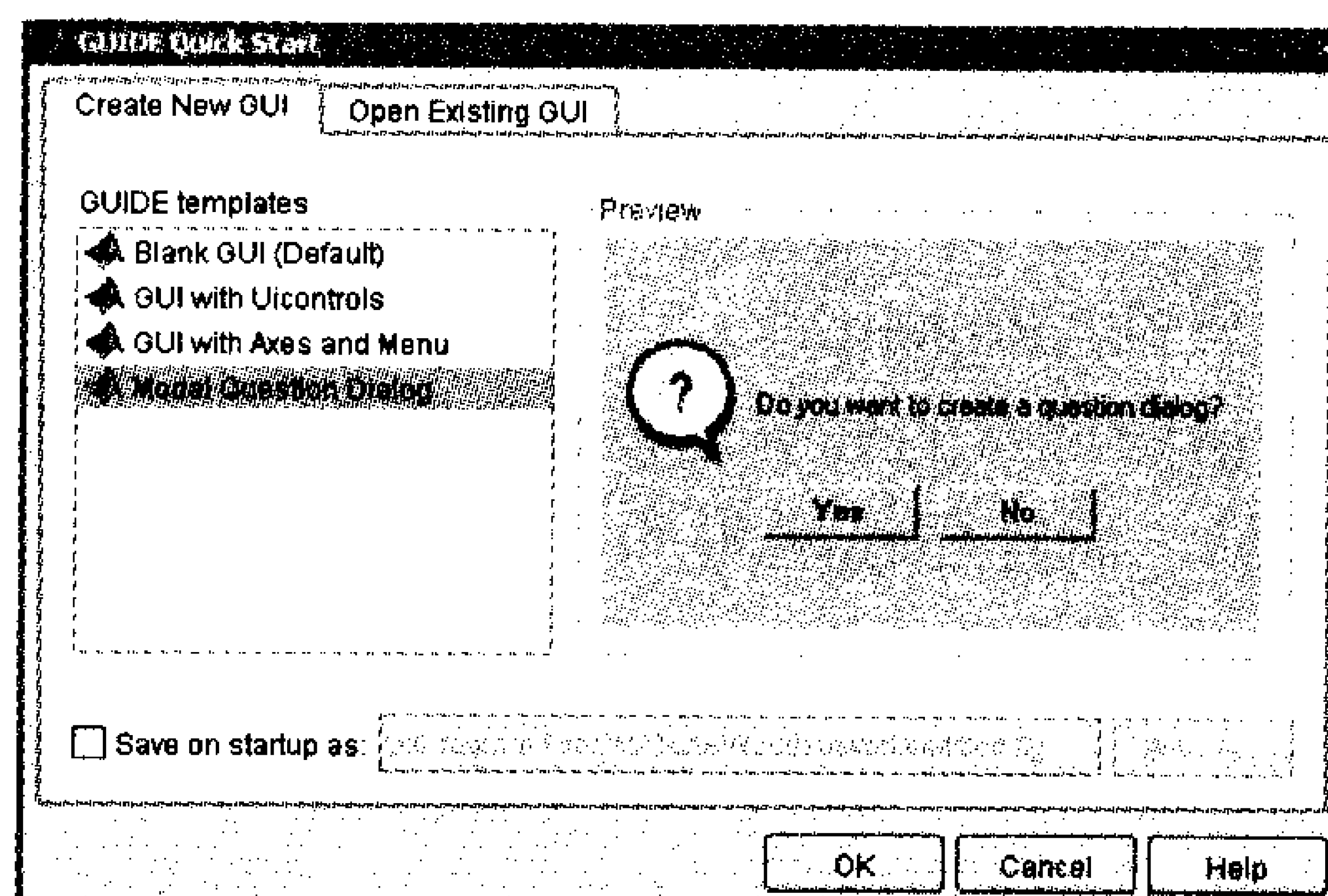


图 9-2 GUIDE 启动界面

4．一个 GUI 程序的例子

本例中设计了一个文本输入框和滑动条，用户可以在文本框中输入数值来改变滑动条的位置，同时，也可以通过直接改变滑动条位置，而其位置参数显示在文本框中。在本例中将了解以下几个方面。

- 如何创建一个图形窗口。
- 如何创建滑动条控件、文本编辑框控件和静态文本标签控件，以及设置与其相关的属性。
- 回调函数的编写方法。

f9-2.m

```
function slider_gui
fh = figure('Position',[250 250 350 350]);              %创建一个图形窗口对象-
%---------------以下使用 uicontrol 函数创建 3 个图形对象控件，并设置各个控件的属性-----------
sh = uicontrol(fh,'Style','slider',...                  %-创建一个滑动条控件
               'Max',100,'Min',0,'Value',25,...
               'SliderStep',[0.05 0.2],...
               'Position',[300 25 20 300],...
               'Callback',@slider_callback);
eth = uicontrol(fh,'Style','edit',...                   %-创建文本编辑框控件
               'String',num2str(get(sh,'Value')),...
               'Position',[30 175 240 20],...
               'Callback',@edittext_callback);          %-指定 Callback 属性为调用 edittext_
callback 子函数
sth = uicontrol(fh,'Style','text',...                   %-创建静态文本标签
               'String','Enter a value or click the slider.',...
               'Position',[30 215 240 20]);
number_errors = 0;
previous_val = 0;
val = 0;
```

%-控件和菜单的创建以及它们属性的设置在 MATLAB GUI 程序设计中非常重要，它们是构成图形界面程序最为常见的两类对象，在 9.4 节和 9.5 节将分别详细介绍界面菜单和控件的使用与相关属性，以及它们属性的设置。

```
%---------滑动条的回调程序，把滑动条的属性值赋值给文本编辑框的 String 属性-----------
 function slider_callback(hObject,eventdata)
      previous_val = val;
      val = get(hObject,'Value');
      set(eth,'String',num2str(val));
      sprintf('You moved the slider %d units.',abs(val- previous_val))
   end
%-------文本编辑框的回调程序，把从文本编辑框得到的数据传递给滑动条，出错时显示错误信
息----
function edittext_callback(hObject,eventdata)
      previous_val = val;
      val = str2double(get(hObject,'String'));
      if isnumeric(val) && length(val) == 1 && ...
         val >= get(sh,'Min') && ...
```

```
            val <= get(sh,'Max')
            set(sh,'Value',val);
            sprintf('You moved the slider %d units.',abs(val - previous_val))
        else
            number_errors = number_errors+1;      % 错误次数累加并显示出来
            set(hObject,'String',...
                ['You have entered an invalid entry ',...
                num2str(number_errors),' times.']);
            val = previous_val;
        end
    end
end
```

% 回调函数的编写是 MATLAB GUI 程序的功能设计非常核心和关键的部分，一个图形界面程序的好坏关键就看回调函数，在 9.6 节中将阐述图形用户界面编程。

执行该程序显示结果如图 9-3 所示。

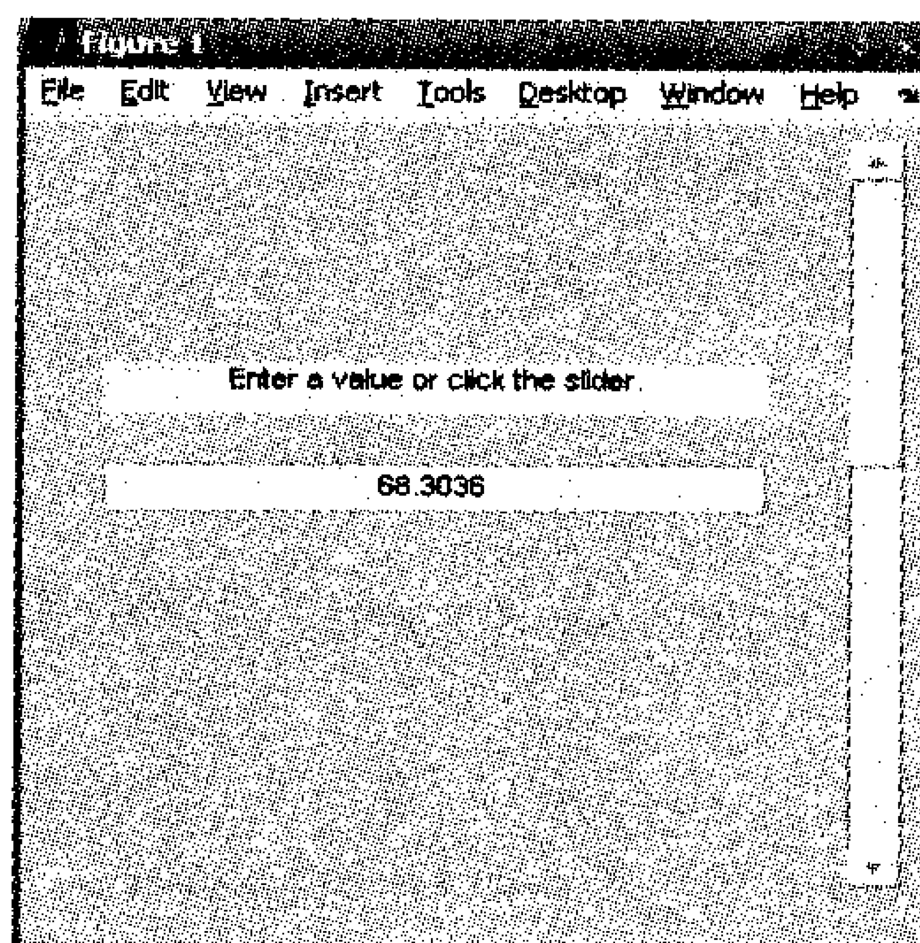


图 9-3　slider_GUI 函数执行结果

9.2　图形用户界面设计工具

在常见的面向对象编程语言中，如 Delphi 和 Visual Basic 等都提供了一个非常简单、方便的图形界面设计工具。在 9.1 节中已经介绍，MATLAB 也提供了一组实用的图形界面设计工具——GUIDE。

MATLAB R2007 提供的 GUI 设计工具主要包括以下几种。

- 对象编辑器（Layout Editor）。
- 对象位置调整工具（Align Objects）。
- 菜单编辑器（Menu Editor）。
- Tab 顺序编辑器（Tab Order Editor）。
- M-file 编辑器（M-file Editor）。
- 对象属性编辑器（Property Inspector）。
- 对象浏览器（Object Browser）。

下面我们对这些 GUI 设计工具进行相关的介绍。

9.2.1　对象编辑器（Layout Editor）

在 MATLAB 命令窗口输入 GUIDE 命令或者在 MATLAB 主窗口执行【File】→【New】→【GUI】命令，将弹出如图 9-2 所示对话框，然后单击【Ok】按钮，就会显示“对象编辑器”窗口，如图 9-4 所示。

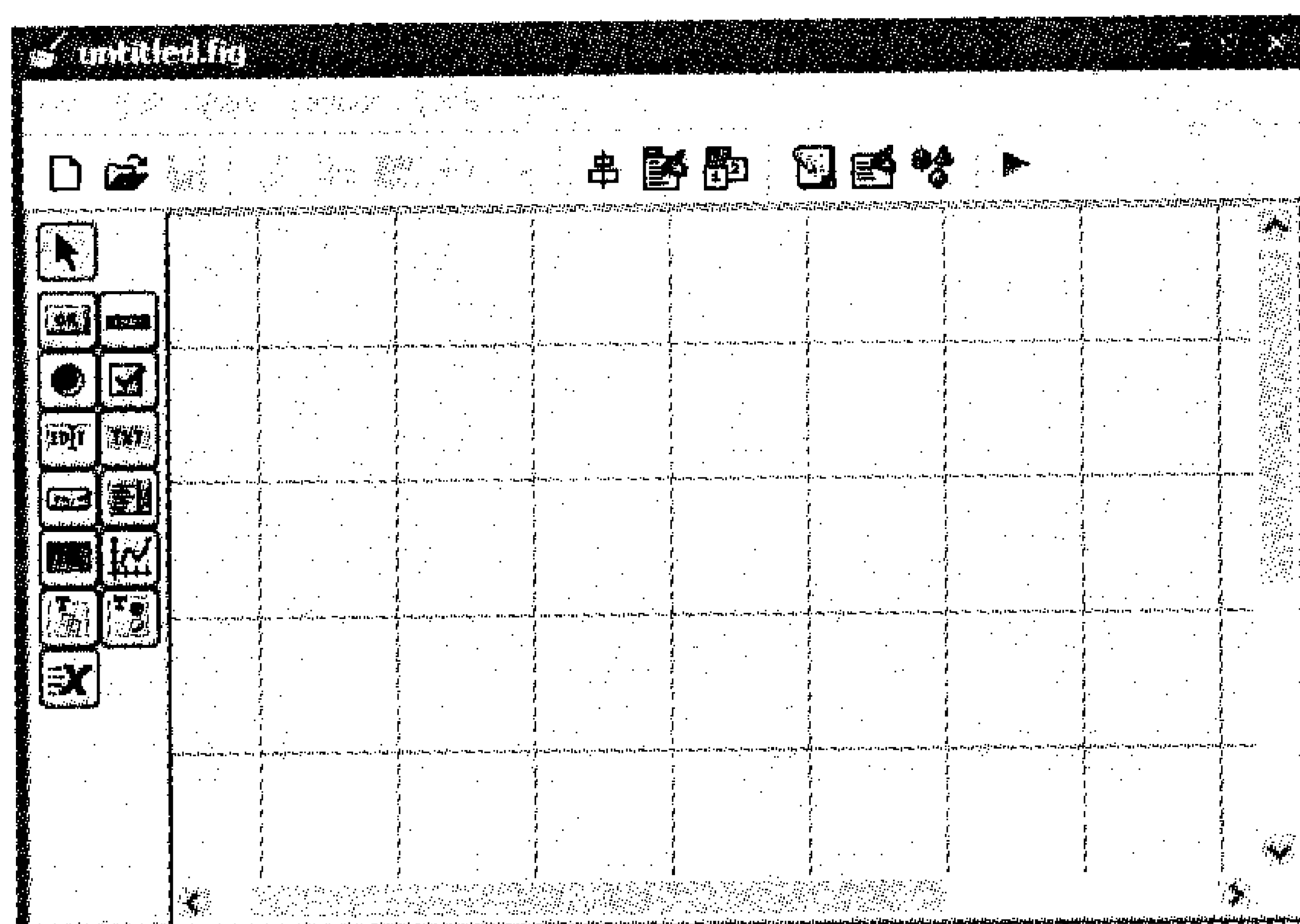


图 9-4　“对象编辑器”窗口

在图 9-4 中，对象编辑器左侧依次是 Push Button（命令按钮）、Slider（滑动条）、Radio Button（单选按钮）、Check Box（复选框）、Text Editor（文本编辑器）、Static Text（静态文本标签）、 Pop-up Menu（弹出菜单）、ListBox（列表框）、Toggle Button（开关按钮）、Panel（框架面板）、 Button Group（按钮群组）、ActiveX Control（ActiveX 控制器）等控件对象和 Axes 坐标轴对象。用户可以根据自己的需求任意选择控件对象或坐标轴对象，如图 9-5 所示为 9.1 节中提到的一个简单的 GUI 程序（见图 9-2）所需的控件和对象。

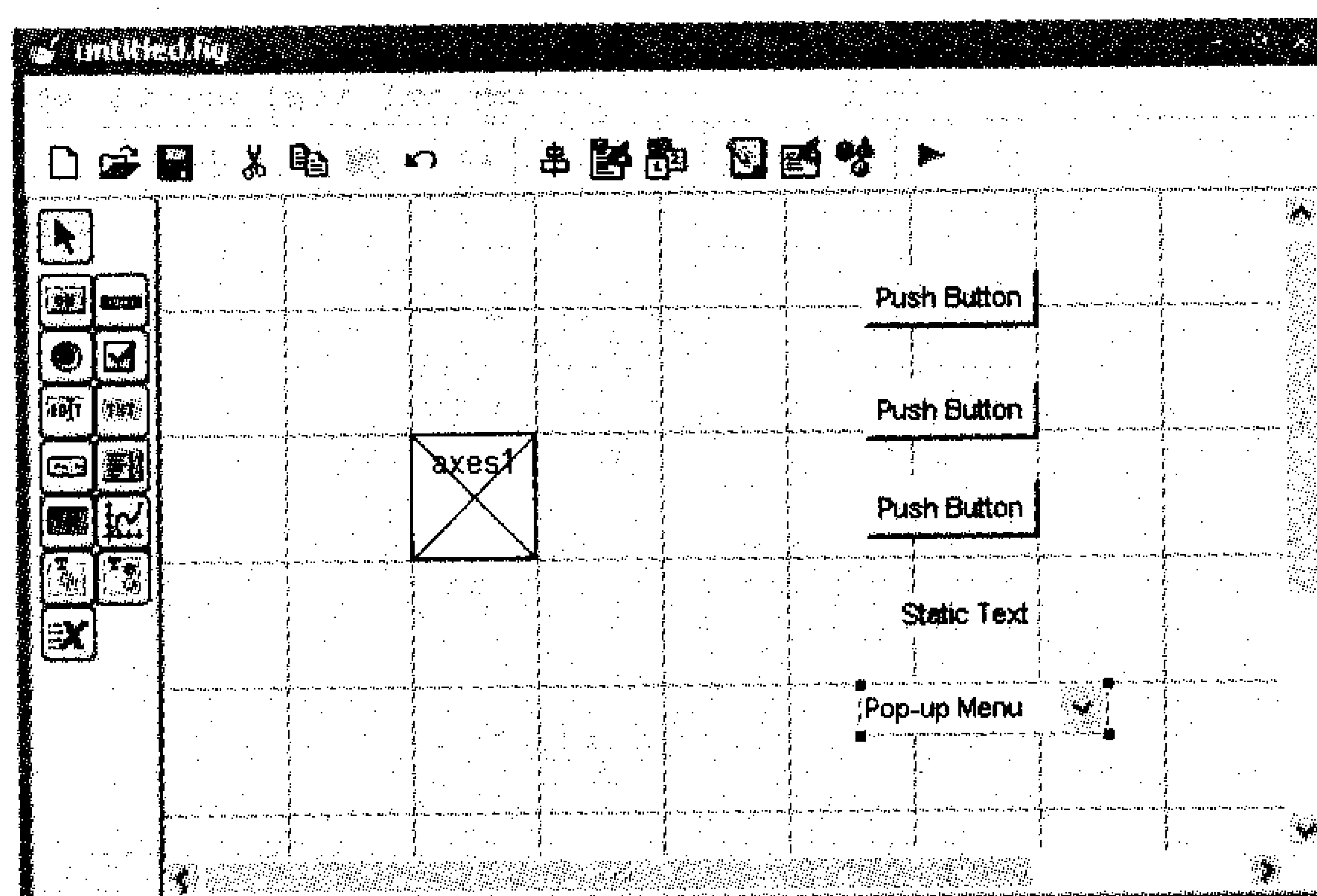


图 9-5　界面设计

在对象编辑器的工具栏上从左至右依次有“对象位置调整工具”、“菜单编辑器”、“Tab 顺序编辑器”、“M-file 编辑器”、“对象属性编辑器”、“对象浏览器”和“激活 GUI 的按钮

（Run）”。通过它们可以更加便捷地设计出想要的 GUI。

当我们选择好界面设计所需的对象以后就可以开始对它们的属性进行相关的设置。例如，在如图 9-6 所示中，用鼠标右键单击【Push Button】按钮，可以从弹出的快捷菜单中选择相应的命令进行设计，单击【M-file Editor】按钮编写相应回调函数。

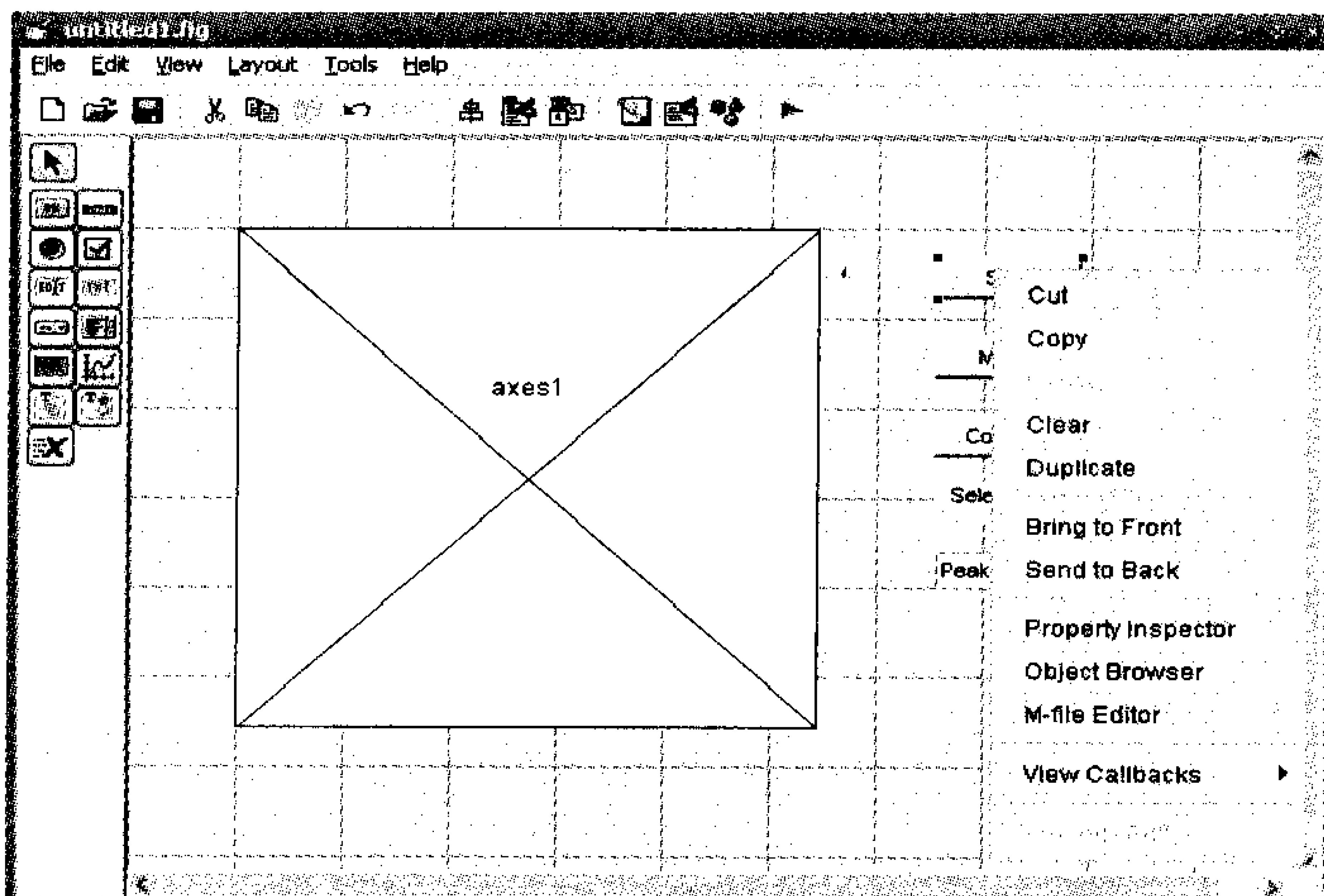


图 9-6 弹出菜单

9.2.2 对象位置调整工具（Align Objects）

在 GUI 设计中，为了设计出来的界面更加美观、规范、统一以及协调，MATLAB 提供了对象位置调整工具。在对象编辑器的工具栏中单击 按钮或者执行【Tool】→【Align Objects】命令，启动以后可以看到如图 9-7 所示的界面。

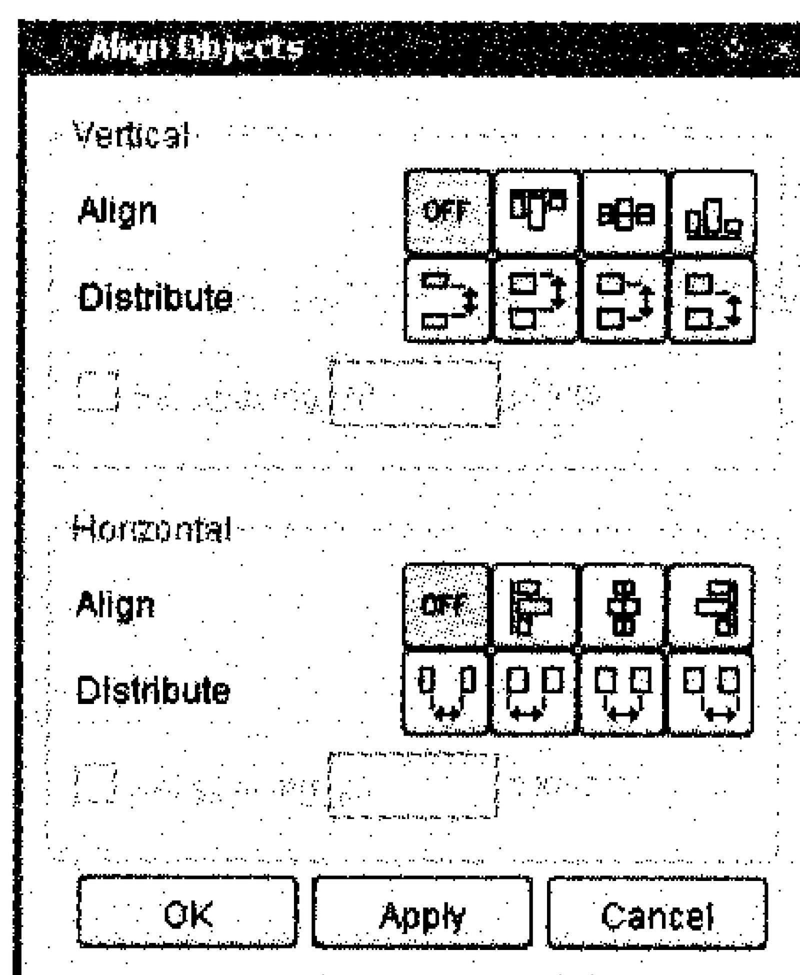


图 9-7 对象位置调整界面

位置调整工具包括垂直位置调整和水平位置调整，选择“Distribute”属性右边的按钮，然后可以在“Set spacing”中调整对象之间的间距，单位为像素。

9.2.3 菜单编辑器（Menu Editor）

在图形界面程序中，菜单操作是最常见的。MATLAB 为用户提供了操作简单的菜单编辑器，在对象编辑器中单击工具栏中的按钮或者执行【Tool】→【Menu Editor】命令，可以弹出如图 9-8 所示的界面。

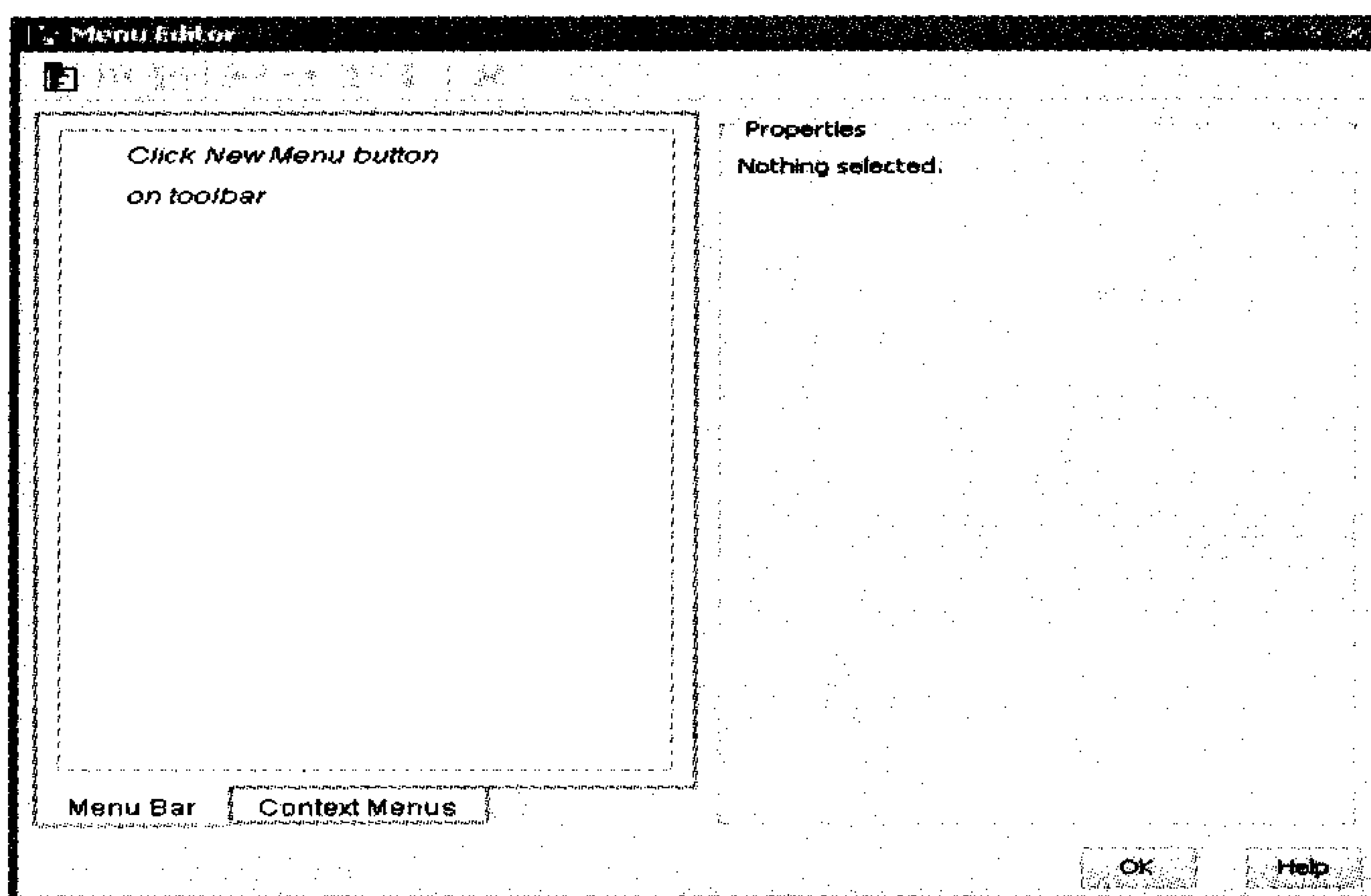


图 9-8 菜单编辑器

单击按钮可以创建下拉式菜单，而单击按钮可以创建下拉菜单的子菜单，左右和上下箭头分别可以改变菜单的级别和上下位置。在菜单编辑器窗口的属性项（UIMenu Properties）可以设置、修改菜单的属性，如图 9-9 所示中的“File”和“Edit”；设置菜单项的快捷键；在“Callback”项可以编写菜单项的回调函数。

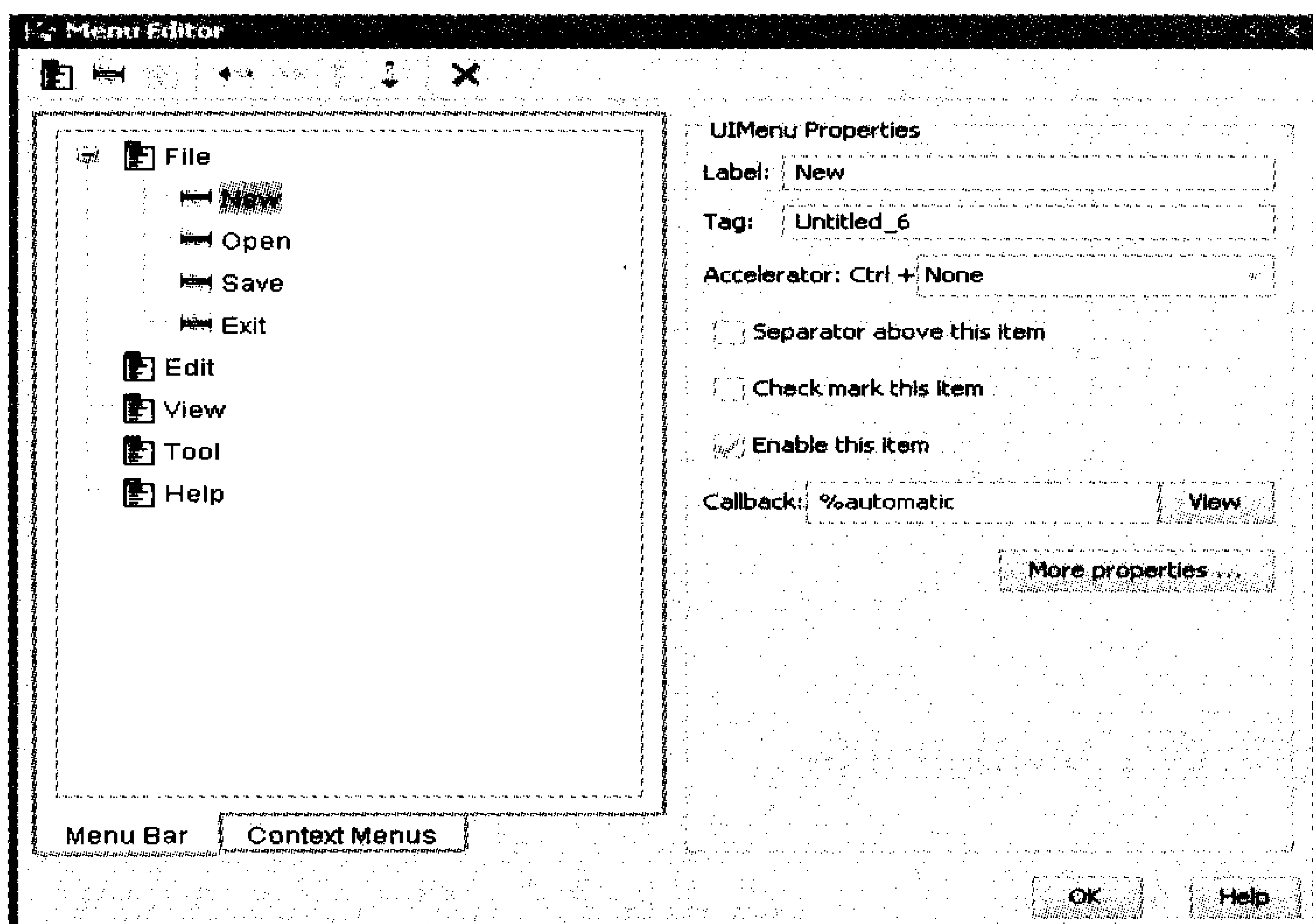


图 9-9 创建下拉菜单和其子菜单

单击菜单编辑器中的“Context Menus”可以创建内容式菜单，也可以修改属性和编写

回调函数，如图 9-10 所示。

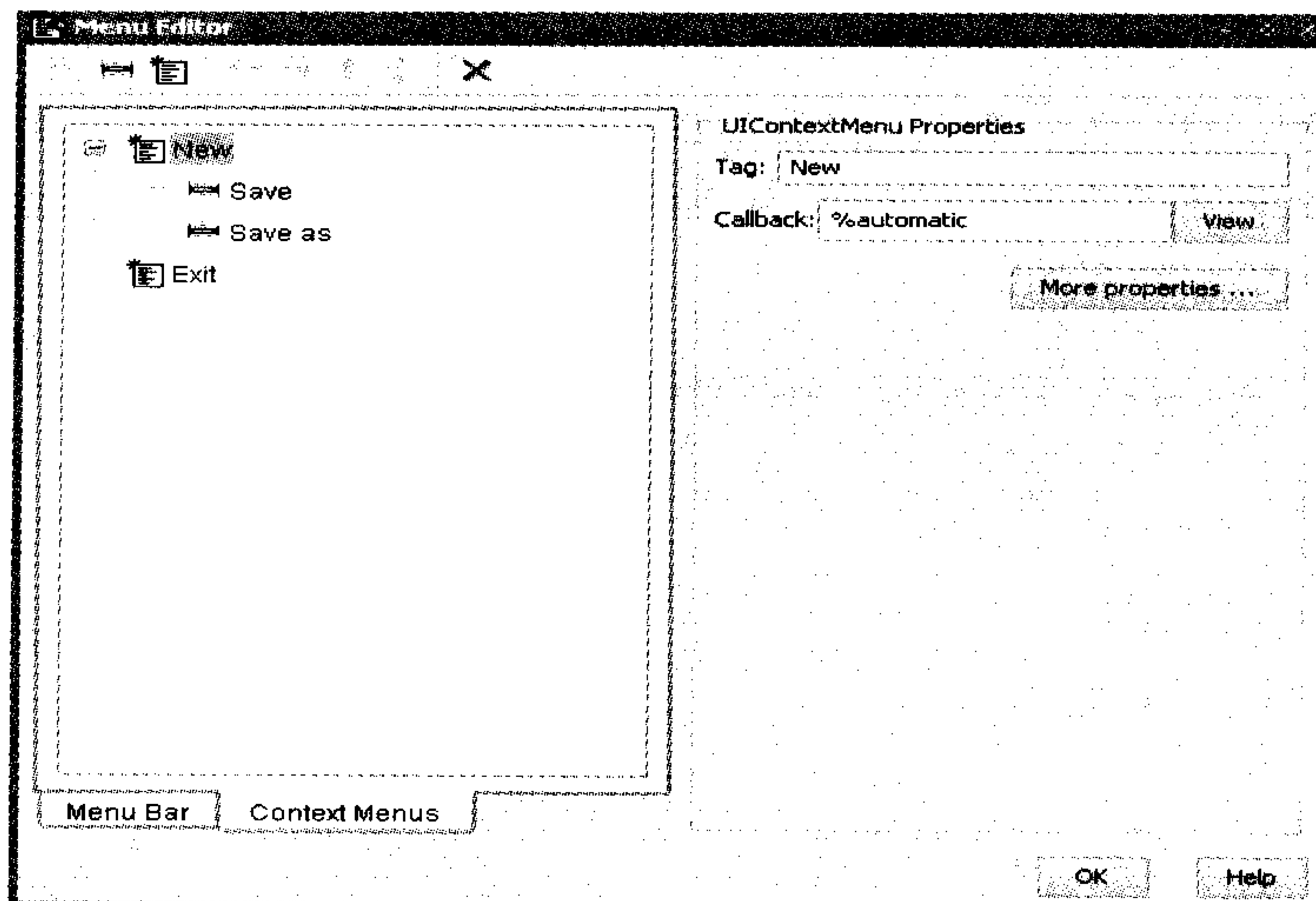


图 9-10　创建内容式菜单

9.2.4　Tab 顺序编辑器（Tab Order Editor）

MATLAB R2007 提供了一个 Tab Order Editor，当需要修改用户按下【Tab】键来改变焦点的顺序时使用该编辑器。单击对象编辑器工具栏中的按钮启动编辑器，如图 9-11 所示。用户通过编辑器界面左上角的上、下角箭头来改变触发焦点的顺序。

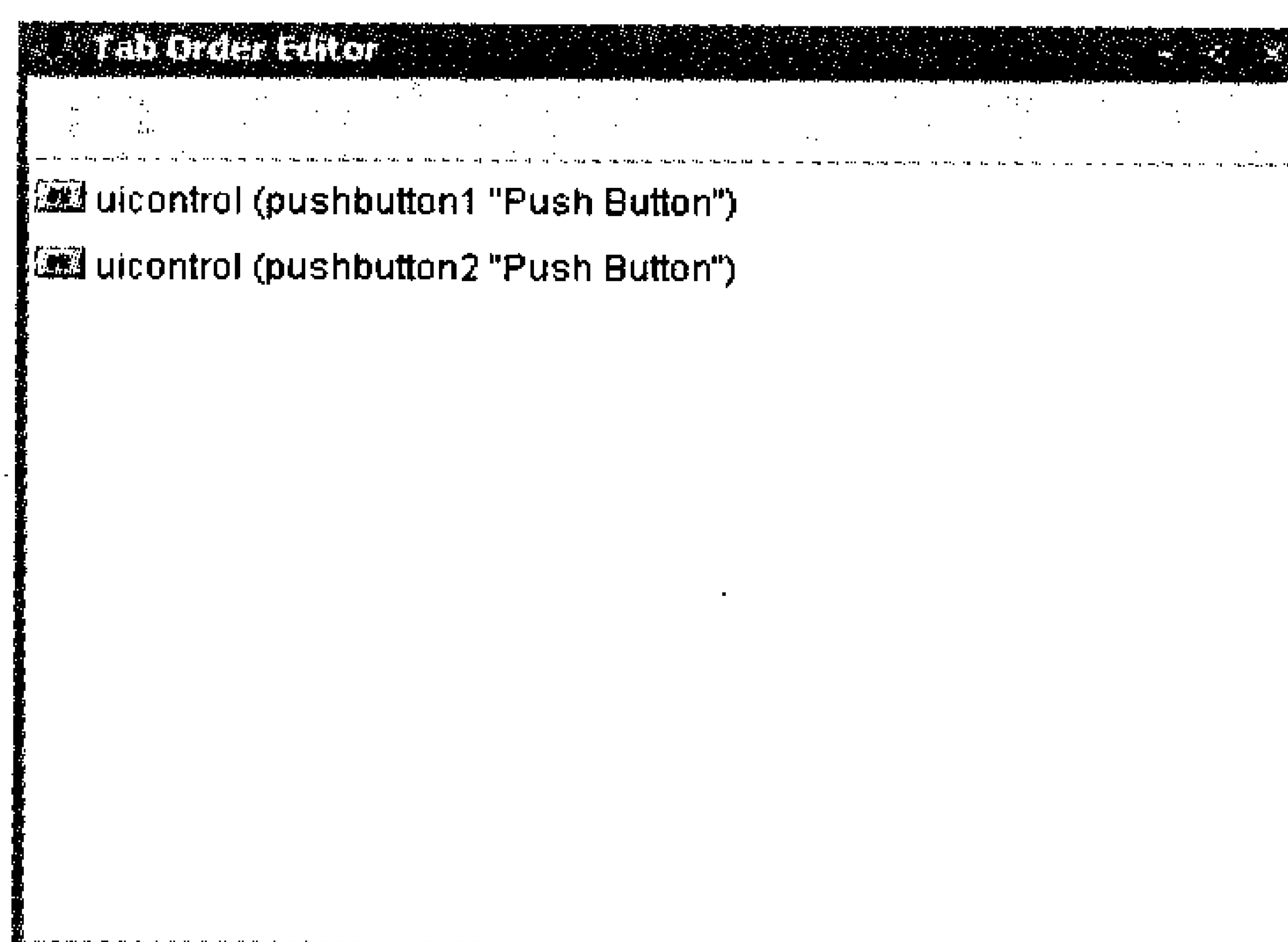


图 9-11　Tab 顺序编辑器

9.2.5　M-file 编辑器（M-file Editor）

在 MATLAB 最近的几个版本中，对象编辑器中提供了 M 文件编辑器的连接。单击对象编辑器工具栏中的按钮或者执行【View】→【M-file Editor】命令启动编辑器，如图 9-12 所示。用户在编辑器中编写自己的回调函数。有关回调函数的编写将在 9.6 节图形用户界面编程中详细阐述。

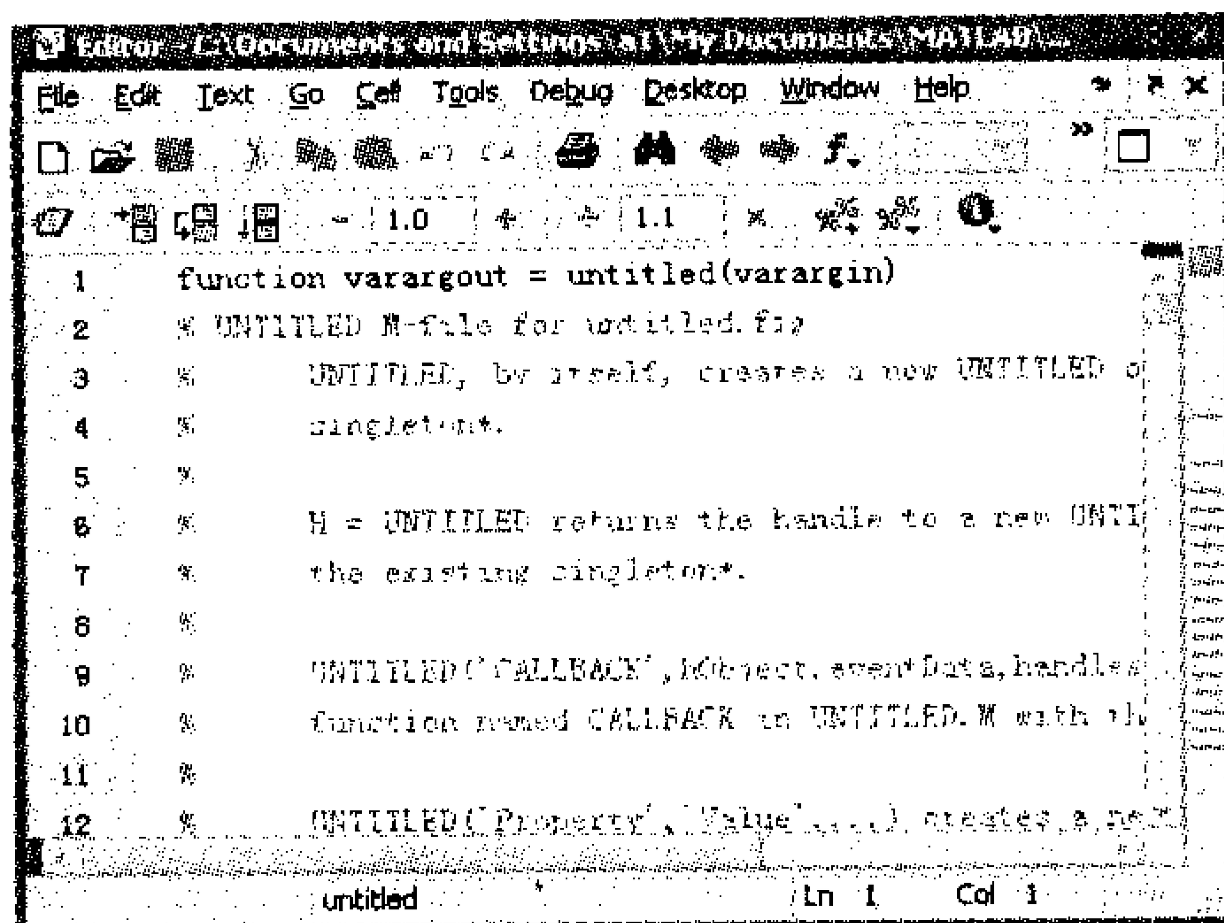


图 9-12 M-file 编辑器

9.2.6 对象属性编辑器（Property Inspector）

MATLAB R2007 提供的对象属性编辑器可以查看、设置和修改每一个对象的属性值。单击对象编辑器工具栏中的按钮、双击选择的对象或者在菜单中执行【View】→【Property Inspector】命令后都可以看到如图 9-13 所示的界面。在设置了对象以后，用户还可以根据自己需要非常方便地查看或修改对象属性。

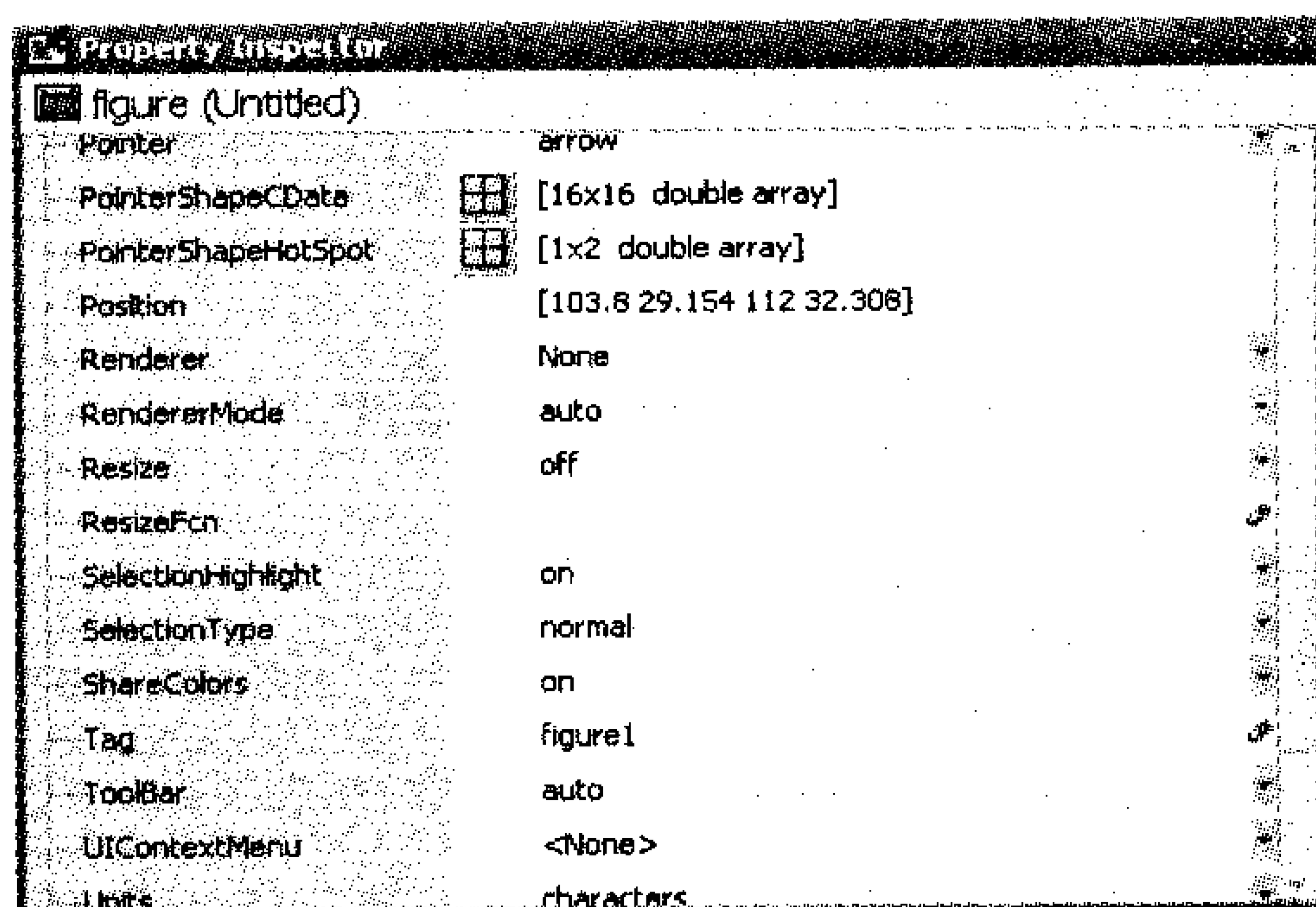


图 9-13 对象属性编辑器界面

9.2.7 对象浏览器（Object Browser）

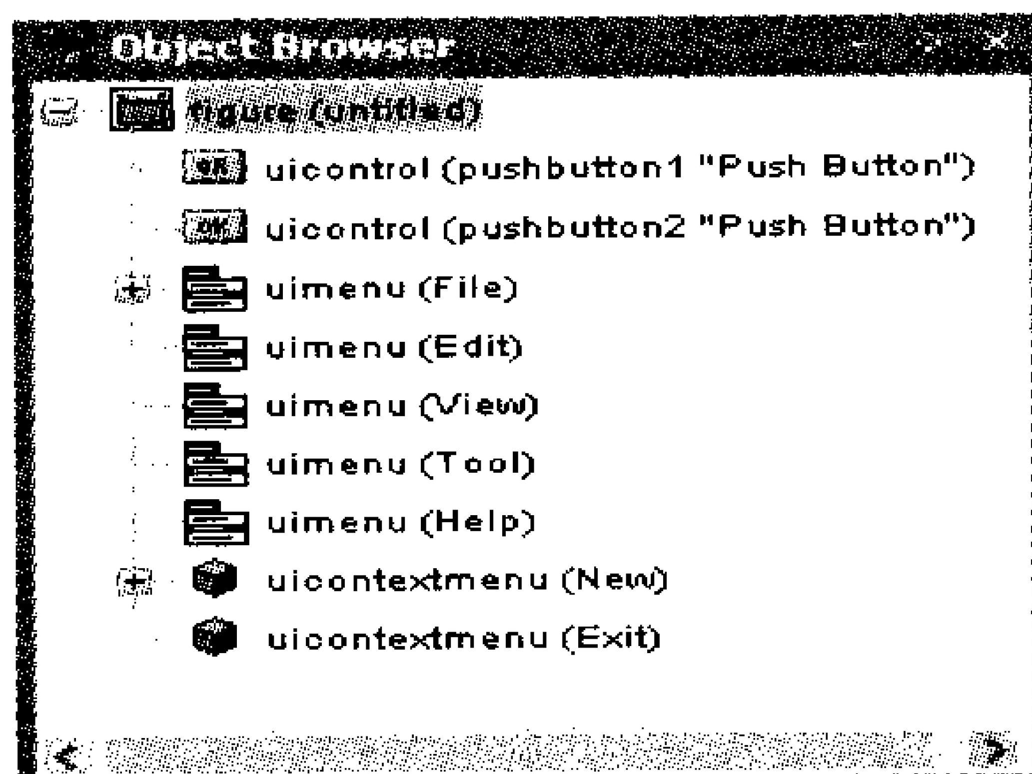


图 9-14 对象浏览器界面

最后介绍的是对象浏览器，单击 按钮或者执行【View】→【Object Browse】命令得到如图 9-14 所示的界面。在对象浏览器中可以看到已在对象编辑器中添加的图形对象，比如，在图 9-14 中就可以看到已经创建的 9 个对象和图形窗口对象 Figure 本身。同时，双击其中任意一个对象可以启动对象属性编辑器。至此我们了解了全部的 GUI 设计工具，单击对象编辑器工具栏中的 ▸ 按钮，这时会得到一个保存提示，单击【是】按钮保存好以后就可以得到 GUI 设计的效果图，如图 9-15 所示。

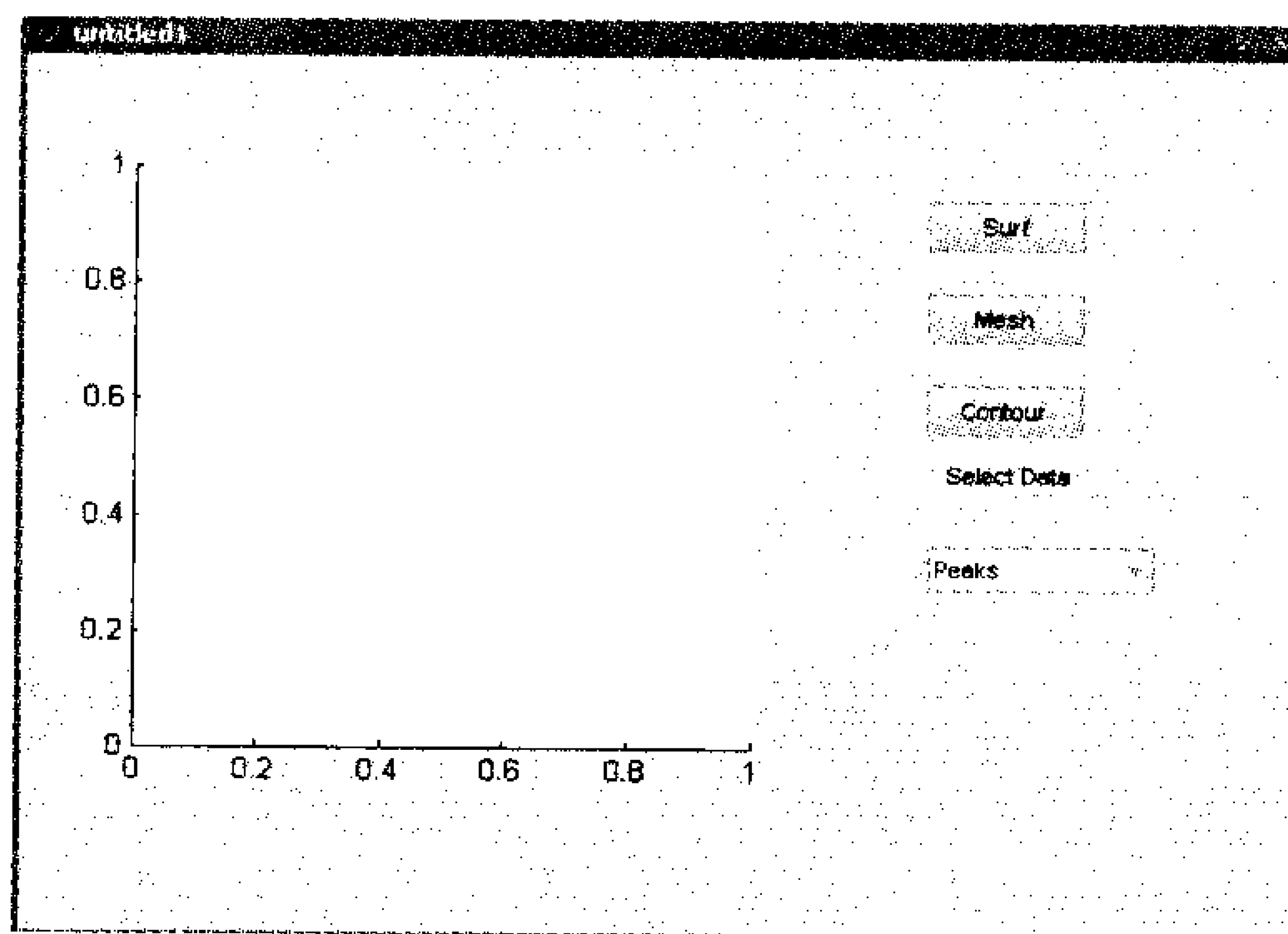


图 9-15　GUI 的外观设计效果

通过 MATLAB 提供的以上 GUI 设计工具，用户可以非常便捷地设计出操作简单、界面友好的图形界面，再加上编写各个对象的回调函数就可以设计出功能强大的 GUI 程序了。

9.3　对话框

对话框是图形用户界面程序常见的对象，也是用户和计算机交互最常见的对象。通过它程序显示相关的系统信息和获取用户数据信息。在常见的面向对象编程的程序语言中，如 Delphi、Visual Basic 等都能够方便地设计对话框。MATLAB R2007 中提供了一系列的对话框函数，其中主要包括公共对话框和一般对话框两大类。常见的函数有 Dialog()、errordlg()、helpdlg()、inputdlg()、listdlg()、msgbox()、Printdlg()、printpreview()、questdlg()、uigetdir()、uigetfile()、waitbar()、uigetpref()、uiopen()、uiputfile()、Uisave()、uisetcolor()、uisetfont()和 warndlg()，等等。下面对 MATLAB 程序设计中常见的对话框进行相关介绍。

9.3.1　公共对话框

1．文件打开对话框

文件打开对话框是 Windows 系统中最常见的对话框之一，在 MATLAB R2007 中设计文件打开对话框调用的函数是 Uigetfile()。

函数格式如下：

```
Uigetfile
[FileName,PathName,FilterIndex]=uigetfile(FilterSpec)
[FileName,PathName,FilterIndex]=uigetfile(FilterSpec,DialogTitle)
```

[FileName,PathName,FilterIndex]=uigetfile(FilterSpec,DialogTitle,)
[FileName,PathName,FilterIndex]=uigetfile(...,'MultiSelect',selectmode)

用户可以根据需要设置不同的打开属性，如用 uigetfile(FilterSpec)格式打开，则显示文件打开对话框中，列出的是默认路径下的由“FilterSpec”属性指定的文件类型，如“FilterSpec”指定的文件类型为“.m”，则显示的为 MATLAB 的 M 文件，如图 9-16 所示。参数 DialogTitle 可以设计对话框的标题，参数 DefaultName 为打开一个默认的文件，参数 selectmode 为打开选择模式，表明是打开一个还是多个文件。

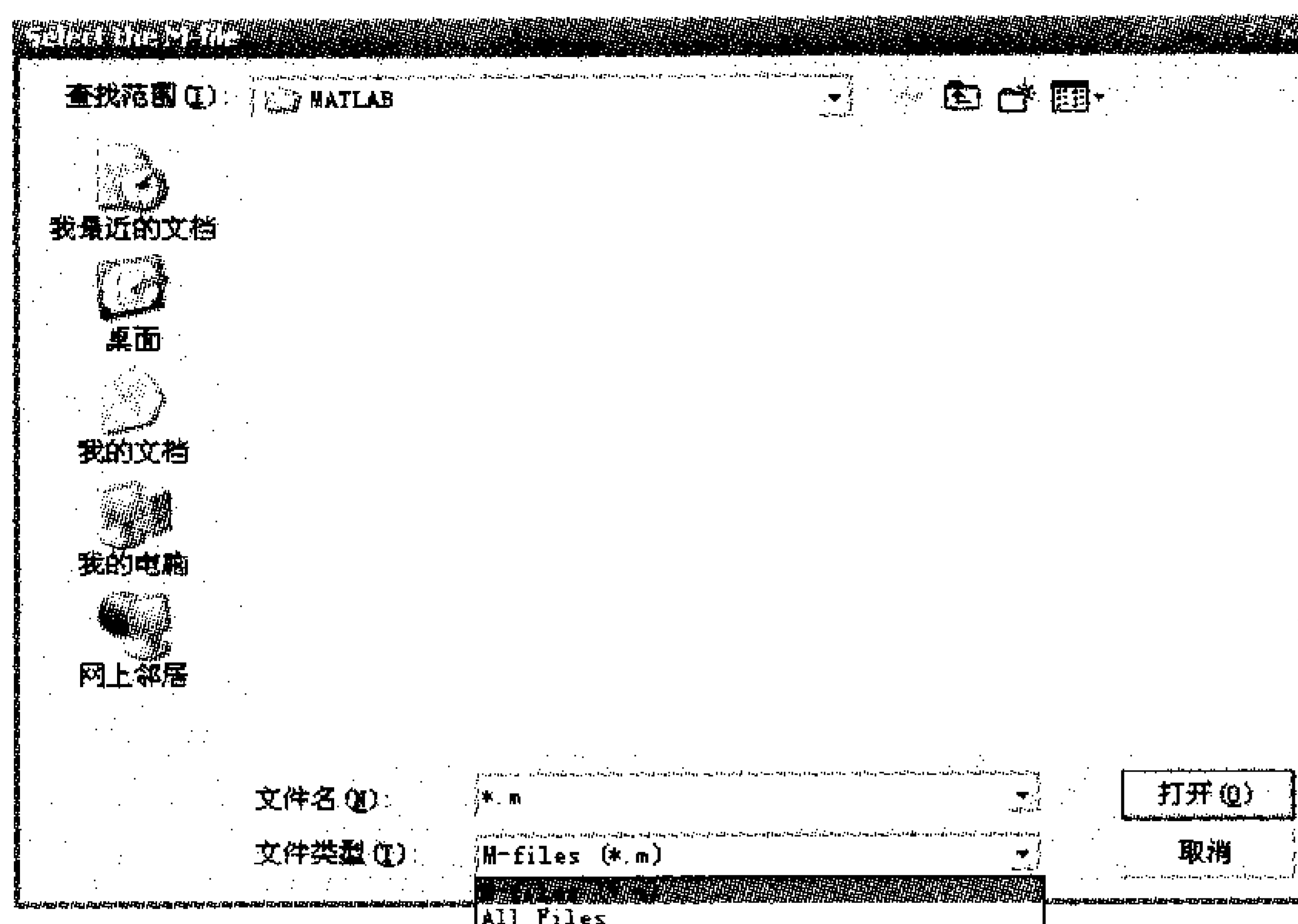


图 9-16 文件“打开”对话框

2. 文件保存对话框

文件保存对话框也是在应用软件应用最广泛对话框之一，它的调用函数为 Uiputfile()。

函数格式如下：

uiputfile
uiputfile[FileName,PathName,FilterIndex]=uiputfile(FilterSpec)
[FileName,PathName,FilterIndex]=uiputfile(FilterSpec,DialogTitle)
[FileName,PathName,FilterIndex]=uiputfile(FilterSpec,DialogTitle,DefaultName)

同文件打开对话框一样，文件保存对话框也有指定的保存文件类型，设置“保存”对话框的标题名和默认的文件名保存，等等。如图 9-17 选择保存文件的类型。

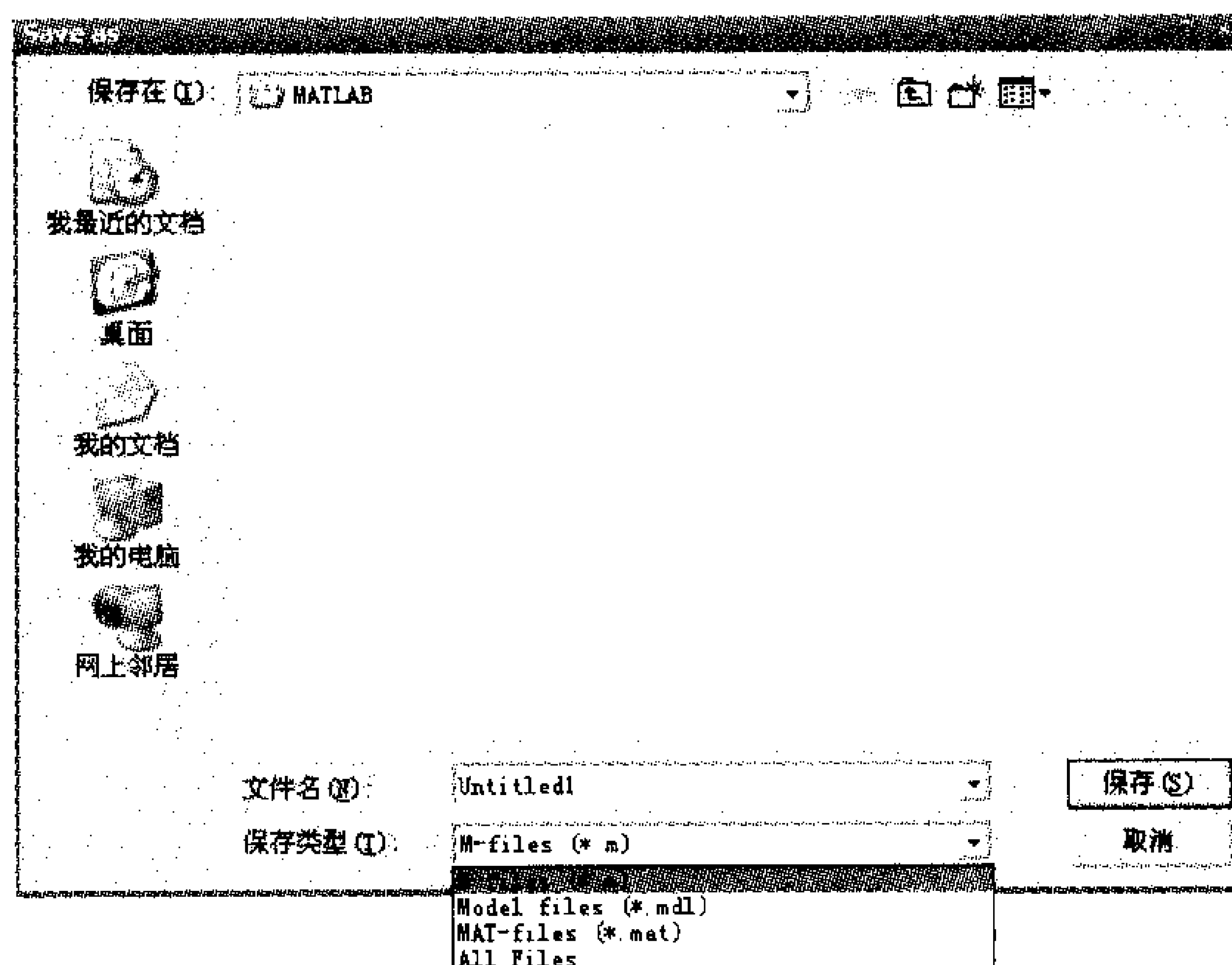


图 9-17 可选择文件类型的“保存”对话框

3．目录选择对话框

目录选择对话框是选择打开指定目录的对话框。用户可以用 Uigetdir()函数来设置打开目录的形式。

函数格式如下：

```
Uigetdir
directory_name = uigetdir
directory_name = uigetdir(start_path)
directory_name = uigetdir(start_path,dialog_title)
```

例如，要打开 MATLAB 的安装目录，在回调函数里面增加代码 uigetdir(matlabroot, 'MATLAB Root Directory')即可，执行就可以得到如图 9-18 所示对话框。

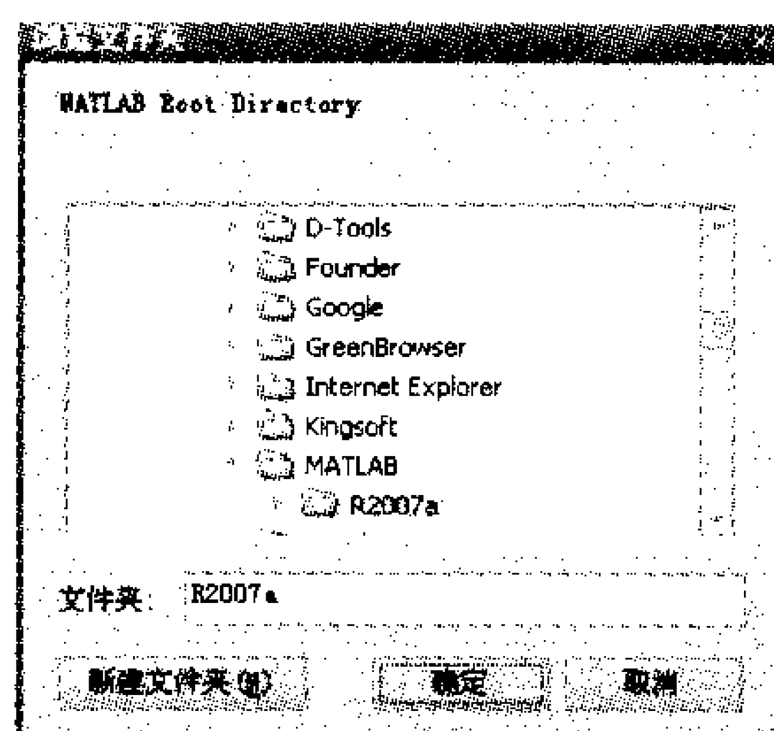


图 9-18 “目录选择”对话框

4．颜色设置对话框

颜色设置对话框用来设置图形对象的背景或者前景颜色，MATLAB R2007 中提供了 Uisetcolor()函数来调用颜色对话框。

函数格式如下：

```
c = uisetcolor          %返回用户选择的颜色，初始值为白色
c = uisetcolor([r g b])    %返回用户选择的颜色,“r” “g” “b”的值为 0 或者 1
c = uisetcolor(h)    %返回用户选择的颜色，h 必须是包含颜色属性的句柄图新搞对象
c = uisetcolor(...,'dialogTitle')   %显示一个特定的标题
```

例 9.1

在 MATLAB 主命令窗口输入：

```
>> c = uisetcolor([1 1 0],'Please select color');
```

出现如图 9-19 所示的“颜色设置”对话框。

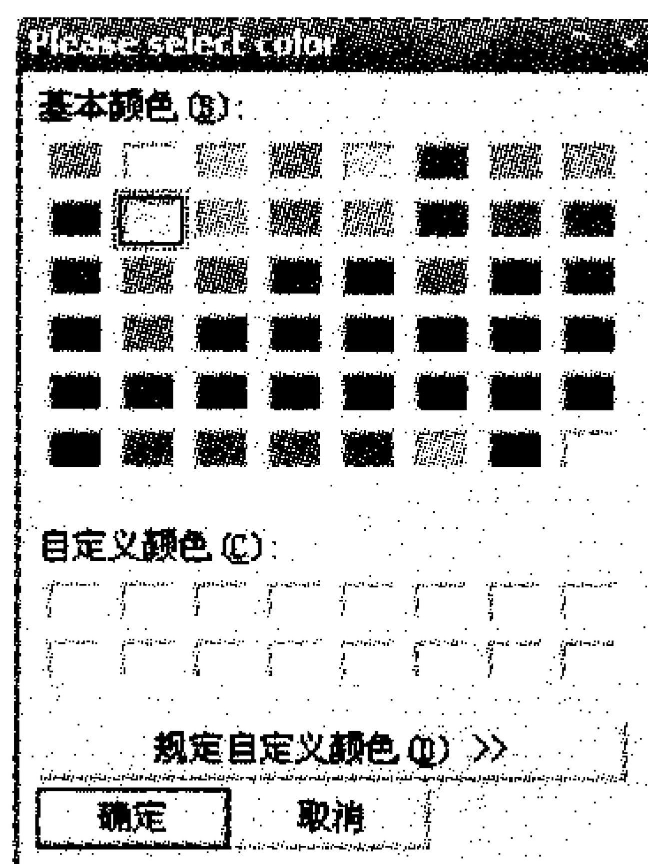


图 9-19 “颜色设置”对话框

单击【取消】按钮，那么主窗口返回的值如下：

```
c =
          1     1     0
```

5．字体设置对话框

字体“设置”对话框在文本编辑软件中比较常见，在 MATLAB 中它用来修改 text、axes 或 uicontrol 对象的字体属性。常见的字体属性有 FontName、FontUnits、FontSize、FontWeight 和 FontAngle。用户可以使用函数 Uisetfont()来调用字体设置对话框。

函数格式如下：

```
uisetfont          % 显示设置对话框，返回用户选择字体属性
uisetfont(h)       % 参数 h 是一个对象句柄
uisetfont(S)       % 参数 S 为字体属性结构
uisetfont(...,'DialogTitle')   %设置对话框的标题
S= uisetfont(...) %返回字体属性值保存在结构 S 中，若选择“取消”或出错输出值为 0
```

例 9.2

在 MATLAB R2007 命令窗口输入：

```
>>h = text(.5,.5,'Figure Annotation');
uisetfont(h,'Update Font')
```

例子中，首先将看到一个字体“设置”对话框，设置好相应的字体，如图 9-20 所示，单击【确定】按钮，将得到改变了字体属性的文本对象，如图 9-21 所示。

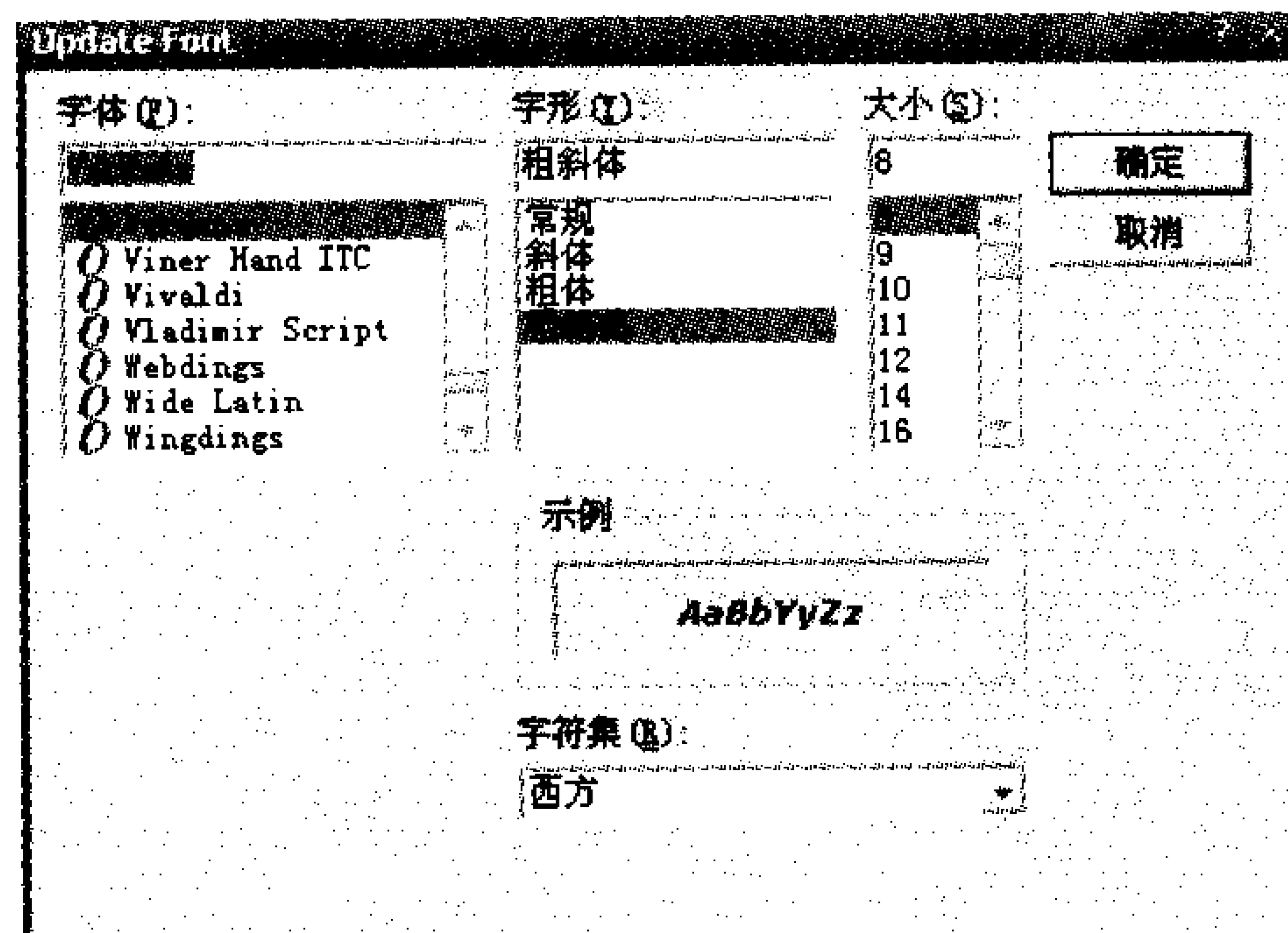


图 9-20　字体“设置”对话框

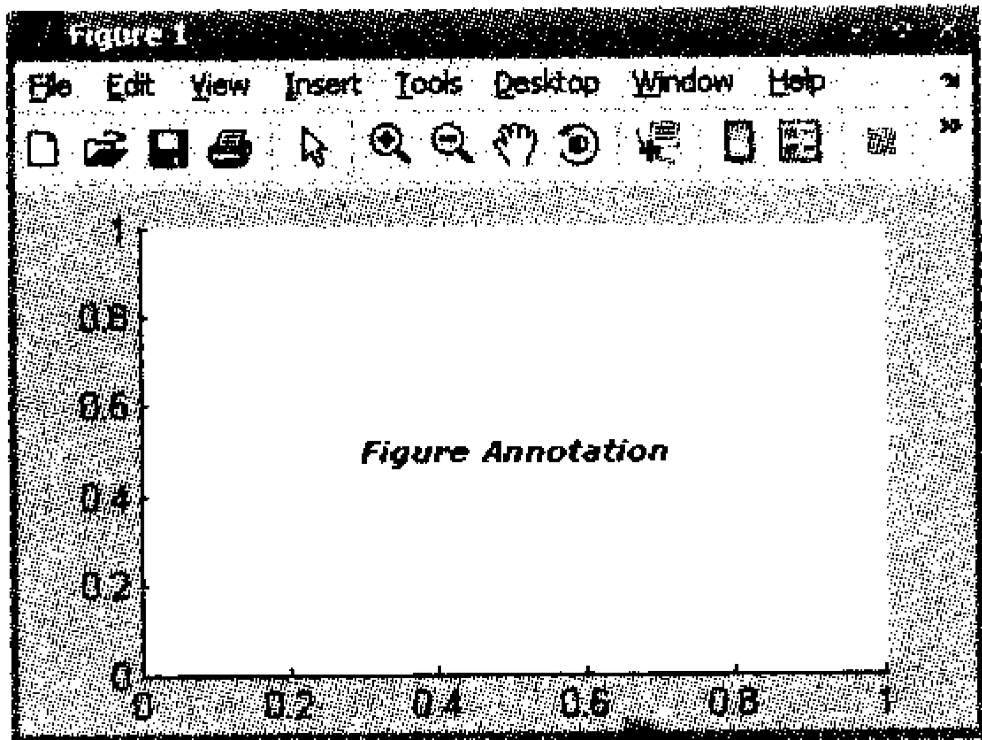


图 9-21　改变字体属性后的文本

然后在命令行窗口输入命令 ans 将得到返回的字体属性值：

```
>> ans
ans =
    FontName: 'Verdana'
    FontUnits: 'points'
    FontSize: 8
    FontWeight: 'bold'
    FontAngle: 'italic'
```

6．打印对话框

“打印”对话框也是应用程序中最为常见的对话框之一。MATLAB R2007 提供了 Printdlg()函数来调用“打印”对话框。

函数格式如下：

```
Printdlg       % 打印当前图形对象
printdlg(fig)  %打印参数 fig 指定图形窗口中的对象，fig 是待打印图形窗口句柄
printdlg('-crossplatform',fig)
%显示标准的 cross-platform MATLAB 打印对话框，fig 是待打印图形窗口句柄
printdlg('-setup',fig)
%显示打印对话框的开始模式，此项可以设置打印的默认选项而不打印
```

例 9.3

```
>>dlg= Printdlg（）;
```

7．打印预览对话框

在文件正式打印前常常可以预览一下打印效果，MATLAB R2007 提供了 Printpreview() 函数来调用“打印预览”对话框。“打印预览”对话框如图 9-22 所示。

函数格式如下：

```
Printpreview   % 显示当前图形窗口对象的打印预览对话框
printpreview(f)   %显示参数 fig 指定图形窗口对象的打印预览对话框
```

例 9.4

```
>> dlg=printpreview;
```

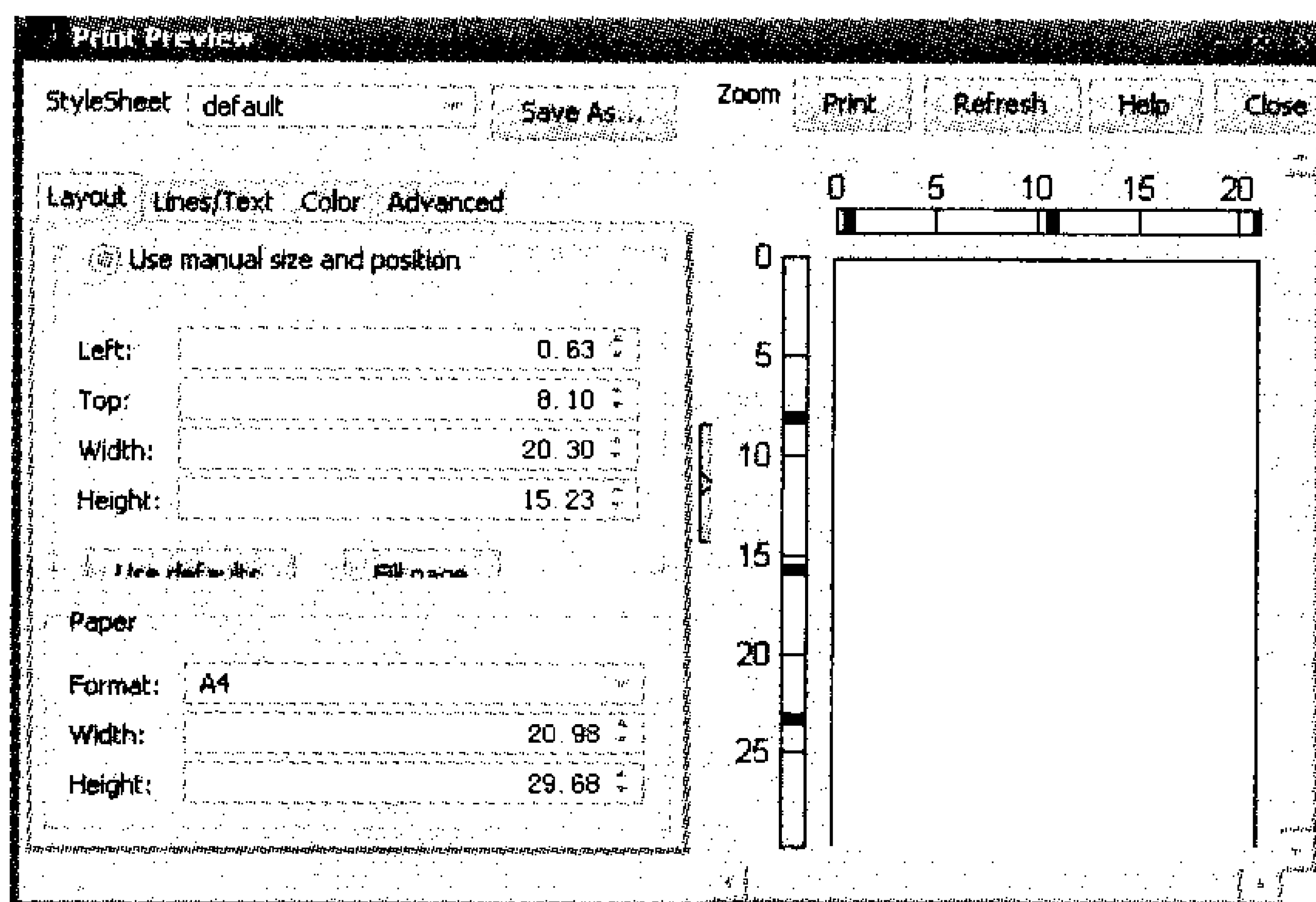


图 9-22 “打印预览”对话框

8．保存工作空间变量对话框

在 MATLAB R2007 中，还可以以 MAT-file 格式保存工作空间的变量，调用函数为 Uisave()。

函数格式如下：

```
uisave
uisave(variables)
uisave(variables,filename)
```

例 9.5

在 MATLAB 命令窗口输入：

```
>> h = 168;
g = 520;
uisave({'h','g'},'TempVar');
```

创建 *h* 和 *g* 两个变量，保存在以“TempVar”为文件名的.MAT 文件里。如图 9-23 所示的变量保存对话框。

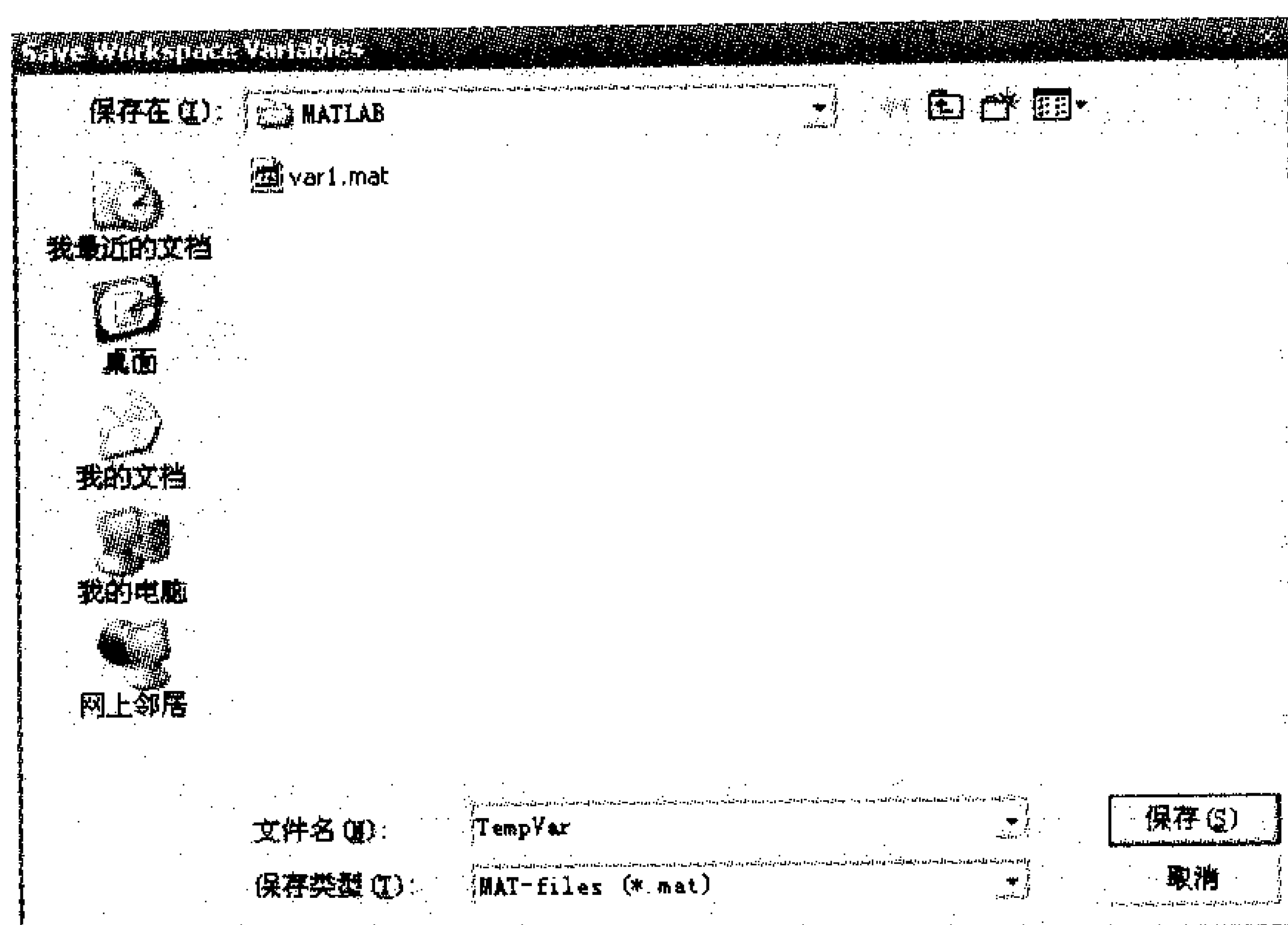


图 9-23 变量保存对话框

9.3.2 一般对话框

介绍完了交互式的公共对话框，将要介绍信息提示类的一般对话框。

1. 错误信息提示对话框

应用软件在用户操作出现错误的时候，一般都会有一个错误信息提示。在 MATLAB 中通过调用 Errordlg()函数来设计错误信息提示对话框。

函数格式如下：

```
h = errordlg          %创建一个默认的错误信息提示框
h = errordlg(errorstring)
                      %创建一个错误信息提示框，提示信息由参数 errorstring 决定
h = errordlg(errorstring,dlgname) %创建一个错误信息提示框，对话框的标题由参数 dlgname 决定，而提示信息由参数 errorstring 决定
h = errordlg(errorstring,dlgname,createmode) %参数 createmode 决定对话框是模式的还是非模式的，它可以是字符串，也可以是结构体。若是字符串的时候值为“modal”“non-modal (default)”“replace”之一
```

例 9.6

```
>> errordlg('File can not open','Open Error');
```

创建的错误提示框如图 9-24 所示。

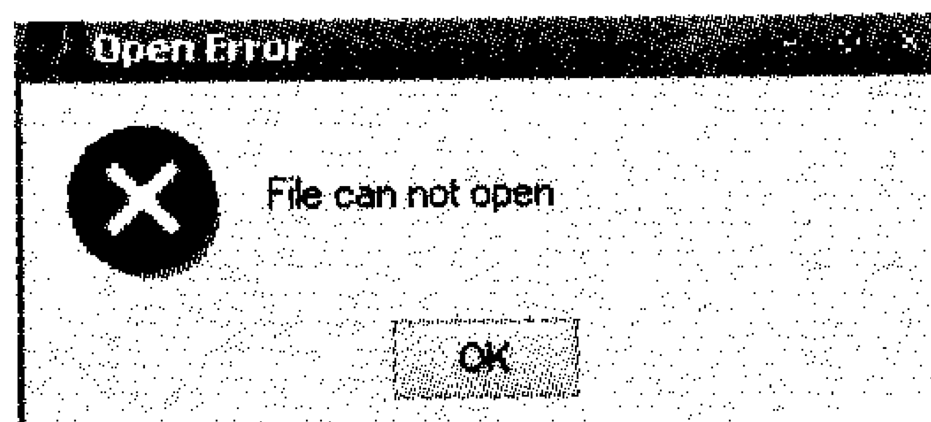


图 9-24 错误提示对话框

2. 帮助对话框

“帮助”对话框在应用程序中的地位越来越高，用户可以在不知如何操作的时候通过该对话框得到信息提示，在 MATLAB 中调用帮助对话框的函数为 Helpdlg()。

函数格式如下：

Helpdlg %创建默认的帮助对话框，默认信息为 This is the default helpstring
helpdlg('helpstring') %创建一个帮助对话框，帮助信息由参数'helpstring'决定
helpdlg('helpstring','dlgname')
%创建一个帮助对话框;帮助信息由参数'helpstring'决定，对话框标题由参数'dlgname'决定
h = helpdlg(...) %返回创建的帮助对话框的句柄，并存放在参数 h 中

例 9.7

```
>> helpdlg('You are right!','TempHelp')
```

创建的错误提示框如图 9-25 所示。

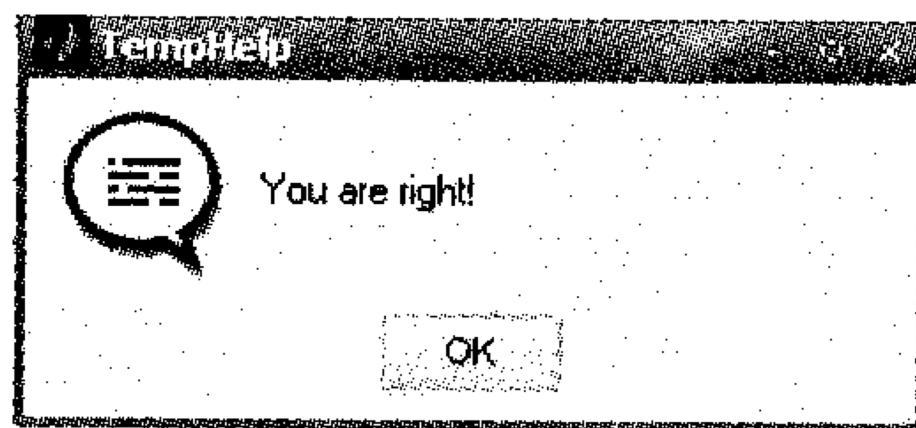


图 9-25 “帮助”对话框

3. 信息提示对话框

信息提示对话框常常是程序设计者在给程序使用者一些必要信息提示或者提醒时设计的一种对话框，调用信息提示对话框的函数为 Msgbox()。

函数格式如下：

h = msgbox(Message) %创建一个默认的提示信息框，显示信息由参数 Message 决定,参数可以是字符串向量，字符串矩阵，或者单元数组。
h = msgbox(Message,Title) %创建一个提示信息框，对话框的标题由 Title 指定
h = msgbox(Message,Title,Icon)
%除了有 Message，Title 外还有独特的图标，由参数 Icon 指定
h = msgbox(Message,Title,'custom',IconData,IconCMap) %对话框的图标可以用户自定义，参数 IconData 存储图像数据,参数 IconCMap 存储图像颜色数据
h = msgbox(...,CreateMode) %参数 CreateMode 决定创建的对话框是模式的还是非模式的。如果是模式的，用户关闭对话框以后控制焦点才从调用的对话框返回；如果是非模式的，用户可以在打开对话框的情况下，将控制焦点从打开的对话框上移开。参数 CreateMode 的值为：“modal”“non-modal (default)”“replace”之一

例 9.8

```
>> h = msgbox('Do not move it','Information','warn') ;
```

创建的信息提示框如图 9-26 所示。

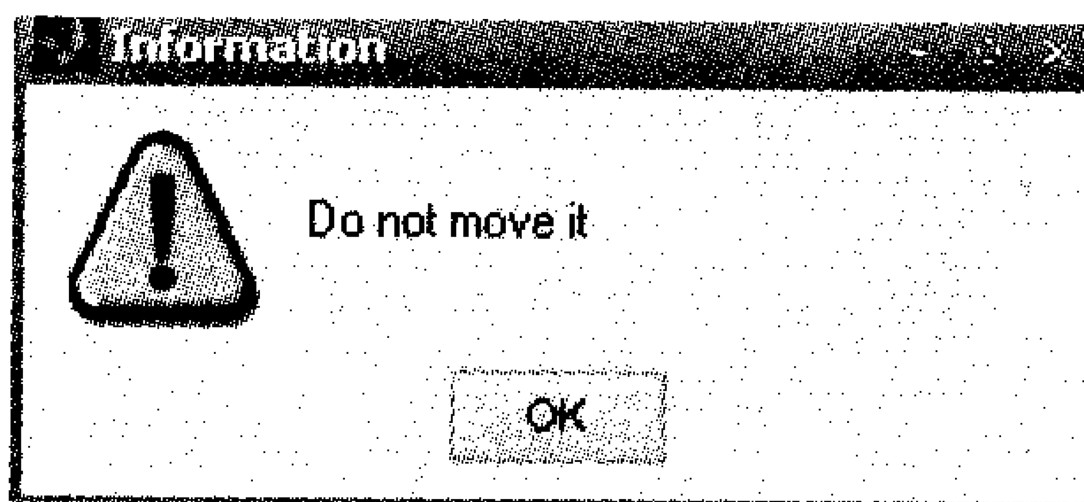


图 9-26 信息提示对话框

4. 警告对话框

一般的应用程序在用户操作错误或者不当的时候，都会出现一个警告对话框来更正用户的操作,在 MATLAB 中调用函数为 Warndlg()。

函数格式如下：

h = warndlg %创建一个默认的警告信息提示框

h = warndlg(warningstring)

%创建一个警告信息提示框，提示信息由参数 warningstring 决定。

h = warndlg(warningstring,dlgname)% 创建一个警告信息提示框，对话框的标题由参数 dlgname 决定，而提示信息由参数 warningstring 决定

h = warndlg(warningstring,dlgname,createmode) %参数 createmode 决定对话框是模式的还是非模式的，它可以是字符串，也可以是结构体。若是字符串的时候值为“modal”“non-modal (default)”“replace”之一

例 9.9

```
>> warndlg('Please pressing OK will exit!','Warning!');
```

结果如图 9-27 所示。

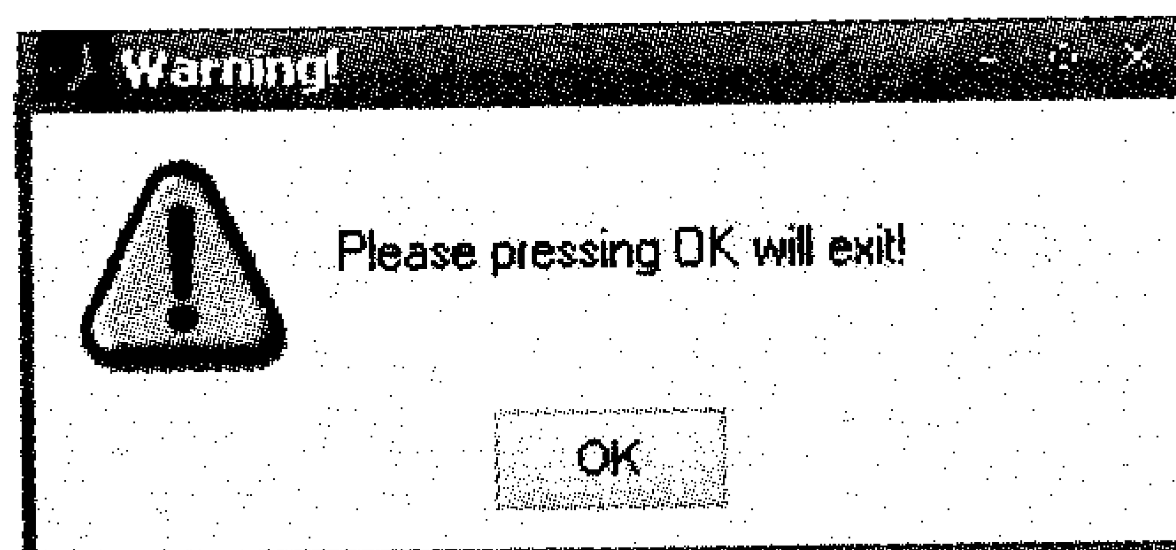


图 9-27 警告对话框

5. 进度条

很多程序都有进度条，如安装程序、下载程序，它用来显示程序的执行进度。MATLAB R2007 提供了函数 Waitbar()用来设计进度条信息框。

函数格式如下：

h = waitbar(x,'message')

%显示一个进度条，参数 message 为提示信息，参数 x 决定对话框显示进度的比例

waitbar(x,'message','CreateCancelBtn','button_callback') %显示的进度条增加一个“Cancel”按钮，当用户按下“Cancel”按钮或者选择退出图形时可以编写相应的回调函数

waitbar(...,property_name,property_value,...)

%可以利用 property_name,property_value 两个可选变量设置进度条的图形属性

waitbar(x) %显示进度条新的进度

waitbar(x,h) %延伸进度条至新的位置

waitbar(x,h,'updated message') %更新进度条的提示信息

例 9.10

```
>> h = waitbar(0.88,'Please wait a while');
```

显示结果如图 9-28 所示。

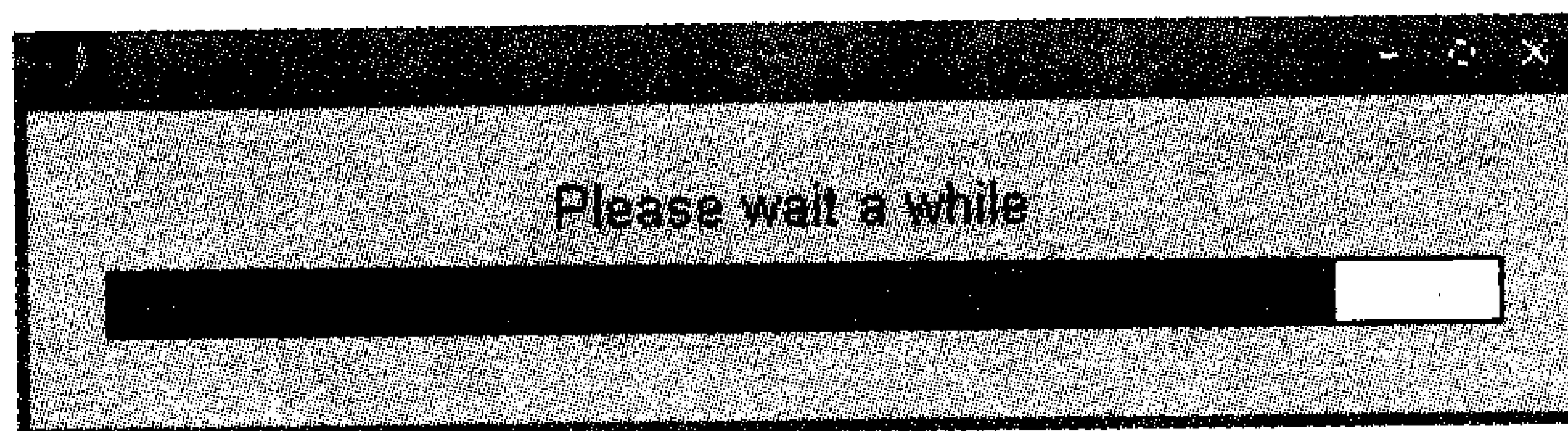


图 9-28 进度条信息框

6．问题显示对话框

当用户面临多种选择的时候，常常会出现一个问题显示对话框来询问用户如何操作。MATLAB R2007 中提供了 Questdlg()函数来创建问题显示对话框。

函数格式如下：

```
button = questdlg('qstring')          %创建一个问题显示对话框，问题由参数 qstring 决定
button = questdlg('qstring','title')       %创建一个问题显示对话框,标题由参数 title 设置
button = questdlg('qstring','title','default')
%返回参数 button 的值由参数 default 设置，default 必须是 'Yes'、'No' 或 'Cancel' 之一
button = questdlg('qstring','title','str1','str2','default') %创建一个有两个命令按钮的问题显示对话框，按钮上的字符窗由'str1'、'str2'决定，返回值 default 为 str1 或 str2
button = questdlg('qstring','title','str1','str2','str3','default')
% 创建一个有 3 个命令按钮的问题显示对话框，按钮上的字符窗由'str1'、'str2'或'str3'决定，返回值 default 为 str1、str2 或 str3 之一
```

例 9.11

```
>> button = questdlg('Do you want exit?','Exit','Yes');
>> button
button =
Yes
```

创建一个"退出"窗口对话框，单击【Yes】按钮，然后再在 MATLAB 窗口输入 Button 命令获得返回参数 Button 值为"Yes"。显示结果如图 9-29 所示。

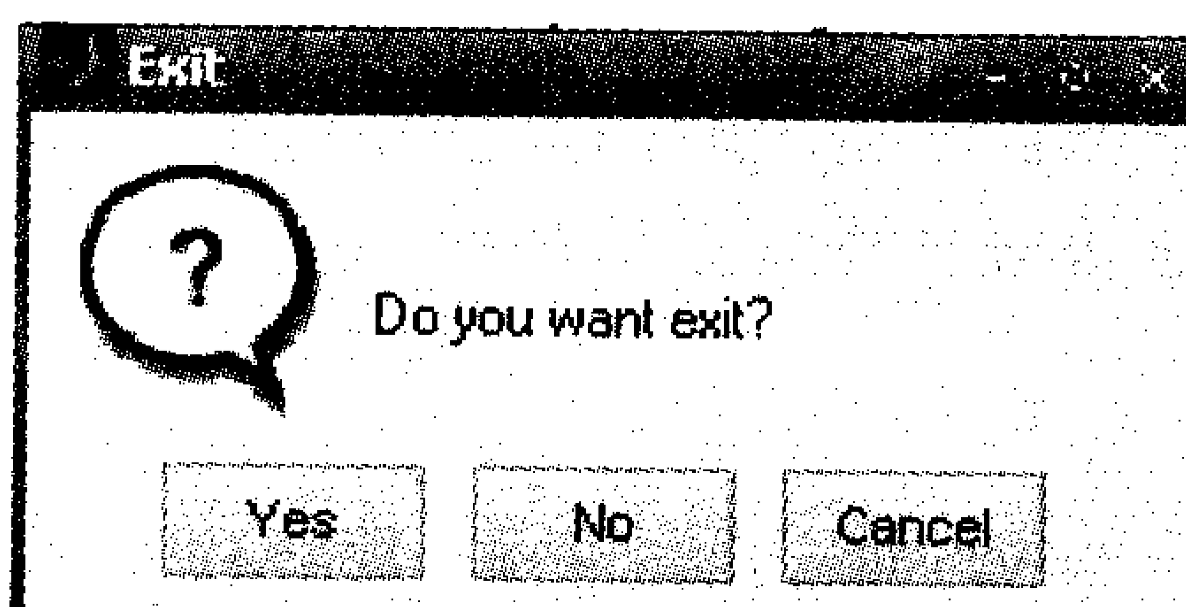


图 9-29　"退出"对话框

7．输入对话框

输入对话框是当计算机程序需要用户输入数据或者变量时显示的对话框。MATLAB R2007 提供了 Uiputfile()函数来创建输入对话框。

函数格式如下：

```
answer = inputdlg(prompt)
% 参数 prompt 为提示信息，用户输入信息存储在返回值 answer 中
answer = inputdlg(prompt,dlg_title)                    %对话框标题由参数 dlg_title 决定
answer = inputdlg(prompt,dlg_title,num_lines)
%对话框里可编辑文本框的行数由参数 num_lines 决定
answer = inputdlg(prompt,dlg_title,num_lines,defAns)
%对话框里可编辑的默认值由参数 defAns 决定
answer = inputdlg(prompt,dlg_title,num_lines,defAns,options)
%对话框的大小是否可以调整由参数 options 决定，options 值为 on 时，对话框可以水平方向调整，options 为结构体时，字段如表 9-1 所示。
```

表 9-1　字段

Field	Description
Resize	Can be 'on' or 'off'(default). If 'on', the window is resizable horizontally
WindowStyle	Can be either 'normal' or 'modal'(default)
Interpreter	Can be either 'none' (default) or 'tex'. If the value is 'tex', the prompt strings are rendered using LaTex

例 9.12

```
>> prompt = {'Enter matrix size:','Enter colormap name:'};
dlg_title = 'Input for peaks function';
num_lines = 1;
def = {'20','hsv'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
```

显示结果如图 9-30 所示。

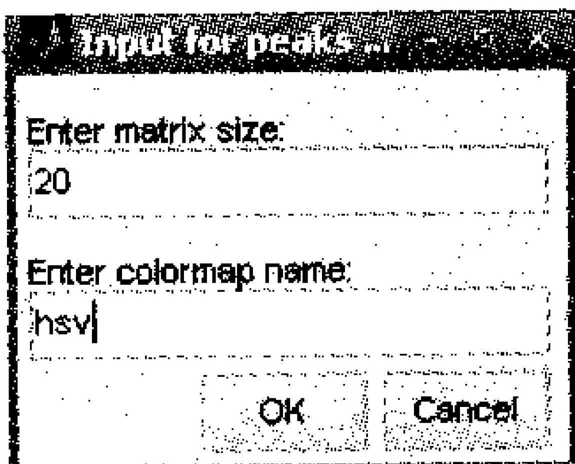


图 9-30　输入对话框

8．选择列表对话框

选择列表对话框是在有多个选项通过列表的形式供用户选择时常常出现的对话框。在 MATLAB R2007 中提供了函数 Listdlg()来创建选择列表对话框。

函数格式如下：

[Selection,ok] = listdlg('ListString',S) %创建一个选择列表模式对话框，用户可以在列表中选择一个或者多个选项

输入参数如表 9-2 所示。

表 9-2　输入参数

Parameter	Description
'ListString'	Cell array of strings that specify the list box items.
'SelectionMode'	String indicating whether one or many items can be selected: 'single' or 'multiple'(the default).
'ListSize'	List box size in pixels, specified as a two-element vector [width height]. Default is [160 300].
'InitialValue'	Vector of indices of the list box items that are initially selected. Default is 1, the first item.
'Name'	String for the dialog box's title. Default is.
'PromptString'	String matrix or cell array of strings that appears as text above the list box Default is {}.
'OKString'	String for the OK button. Default is 'OK'.
'CancelString'	String for the Cancel button. Default is 'Cancel'.
'uh'	Uicontrol button height, in pixels. Default is 18.
'fus'	Frame/uicontrol spacing, in pixels. Default is 8.
'ffs'	Frame/figure spacing, in pixels. Default is 8.

例 9.13

```
>> d = dir;
str = {d.name};
[s,v] = listdlg('PromptString','Select a file:',...
                'SelectionMode','single',...
                'ListString',str);
```

显示结果如图 9-31 所示。

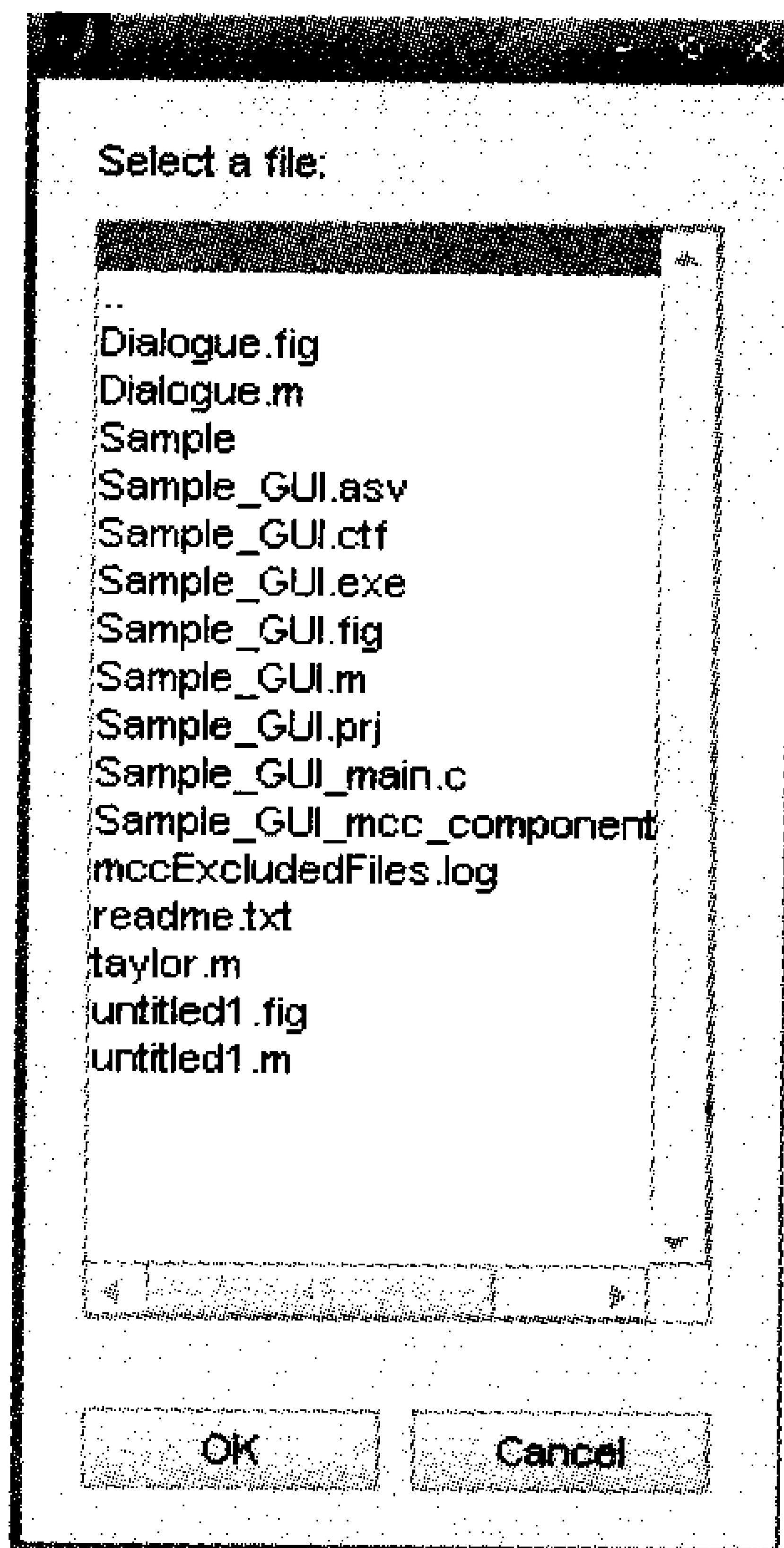


图 9-31　列表选择对话框

9.4　界面菜单

在图形界面的程序中，菜单是常见的操作对象。菜单操作简单明了，用户操作方便、直观。在 MATLAB 中菜单一般分为两种类型：Uimenu 和 Uicontextmenu。下面将对 MATLAB R2007 菜单操作进行相关的介绍。

9.4.1　创建菜单

MATLAB R2007 菜单的创建可以通过 9.2 节介绍的 GUI 设计工具——Menu Editor 或者命令行方式。

1．GUI 设计工具—菜单编辑器（Menu Editor）

正如 9.2 节中所介绍的，利用菜单编辑器中创建菜单非常方便和直观。具体方法请参考 9.2.3 节菜单编辑器（Menu Editor）。用户可以很简单地创建菜单，如图 9-32 所示。

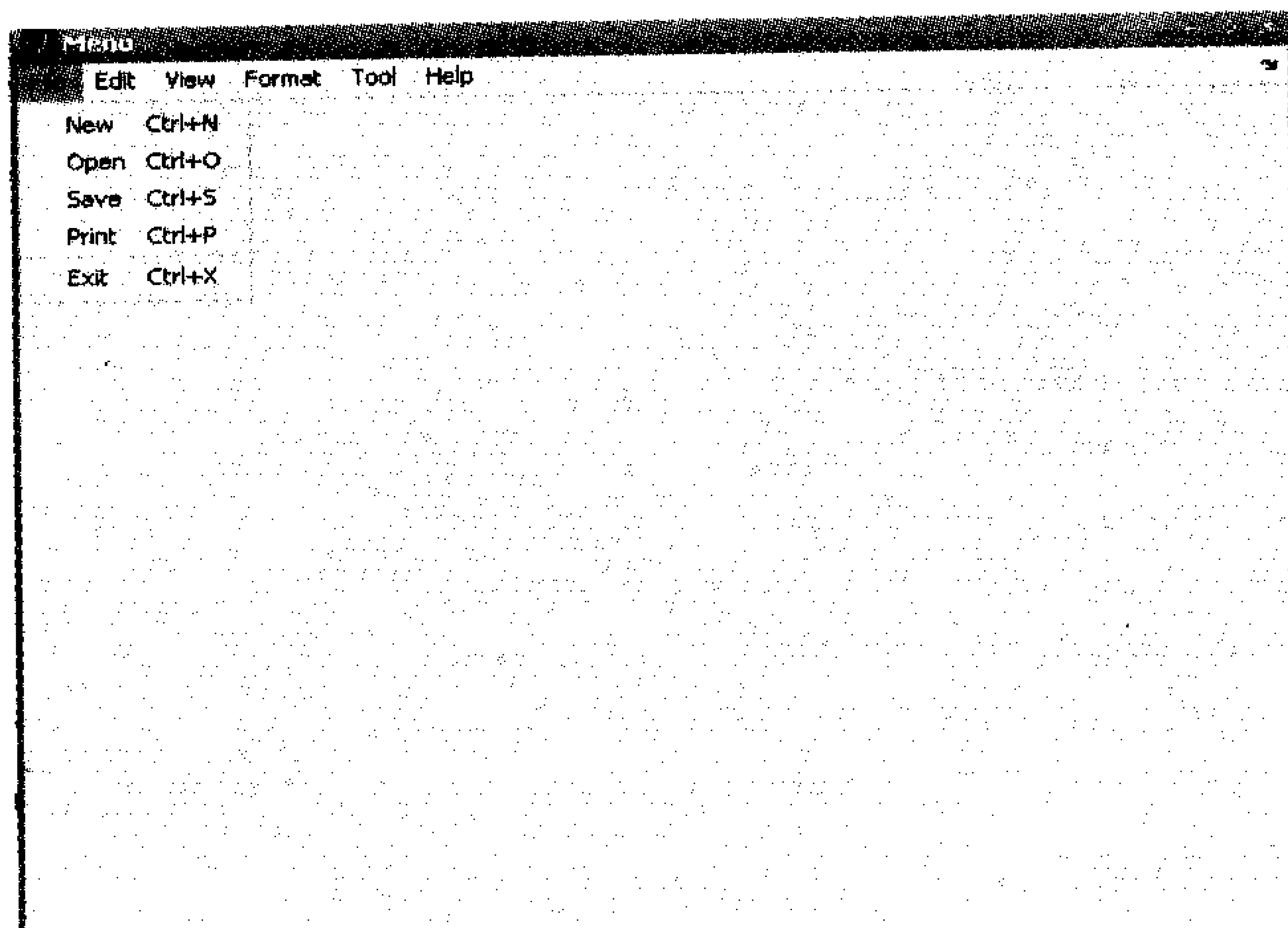


图 9-32　利用 Menu Editor 创建的菜单

2．命令行方式

MATLAB R2007 提供了函数 Uimenu()来创建下拉式菜单。

函数格式如下：

```
handle = uimenu('PropertyName',PropertyValue,...)   %创建下拉菜单，参数 PropertyName 为菜单的属性名，参数 PropertyValue 为属性名对应的属性值
handle = uimenu(parent,'PropertyName',PropertyValue,...)
 %创建一个父菜单名为参数 parent 的子菜单
```

例 9.14

```
>> f = uimenu('Label','Workspace');
      uimenu(f,'Label','New Figure','Callback','figure');
      uimenu(f,'Label','Save','Callback','save');
      uimenu(f,'Label','Quit','Callback','exit',...
               'Separator','on','Accelerator','Q');
```

创建的菜单如图 9-33 所示中【Help】菜单选项后面的。

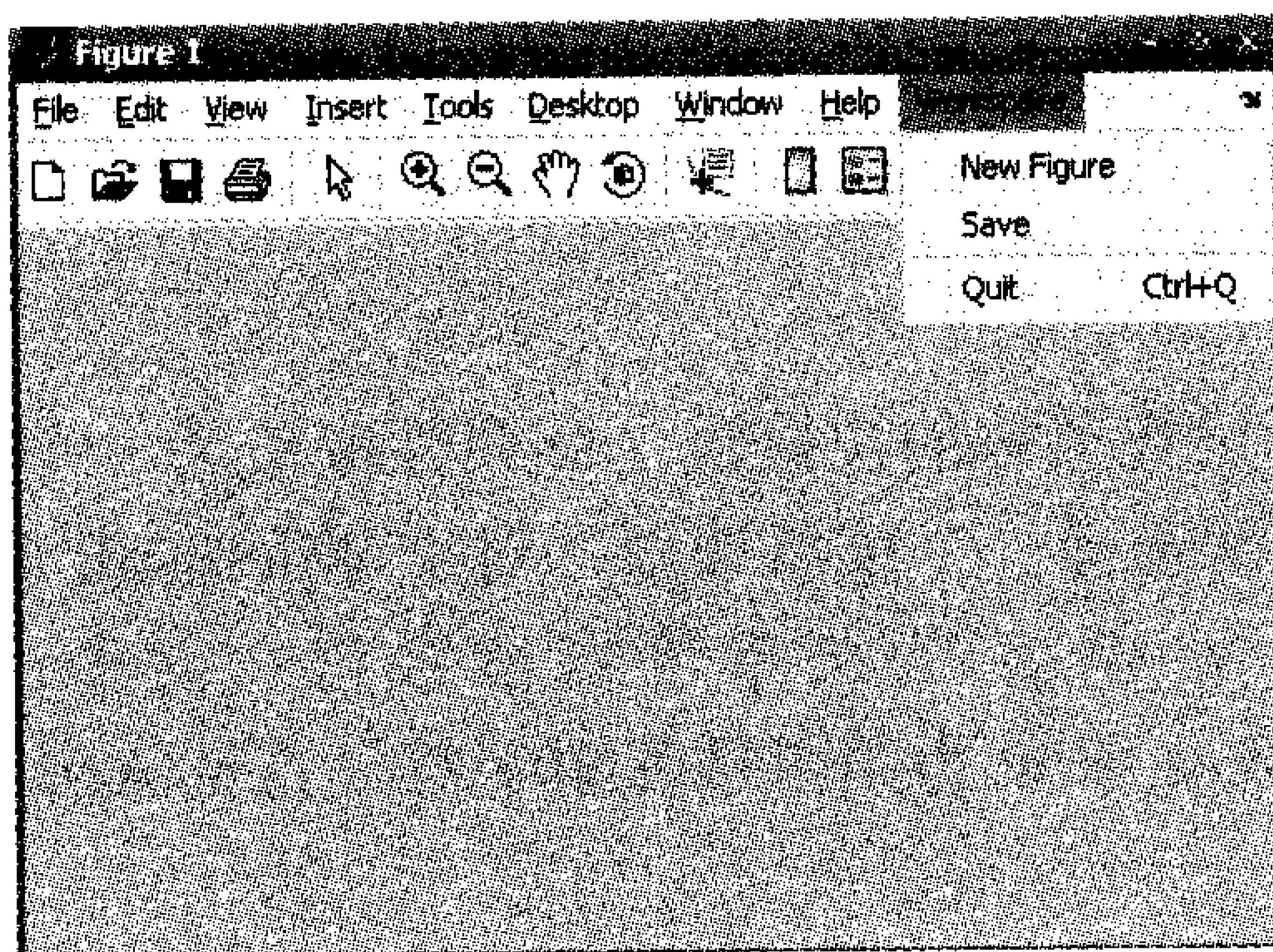


图 9-33　命令行方式创建菜单

MATLAB R2007 还提供了函数 Uicontextmenu()来创建内容式菜单。

函数格式如下：

```
handle = uicontextmenu('PropertyName',PropertyValue,...)
```

例 9.15

```
>>cmenu = uicontextmenu;
hline = plot(1:10, 'UIContextMenu', cmenu);
cb1 = ['set(hline, "LineStyle", "--")'];
cb2 = ['set(hline, "LineStyle", ":")'];
cb3 = ['set(hline, "LineStyle", "-")'];
cb4 = ['set(hline, "LineStyle", "*")'];
item1 = uimenu(cmenu, 'Label', 'dashed', 'Callback', cb1);
item2 = uimenu(cmenu, 'Label', 'dotted', 'Callback', cb2);
item3 = uimenu(cmenu, 'Label', 'solid', 'Callback', cb3);
item4 = uimenu(cmenu, 'Label', 'asterisk', 'Callback', cb4);
```

例子中代码是在图形窗口中画一条直线，然后在直线的附近单击鼠标右键将会出现创建的内容菜单，如图 9-34 所示。

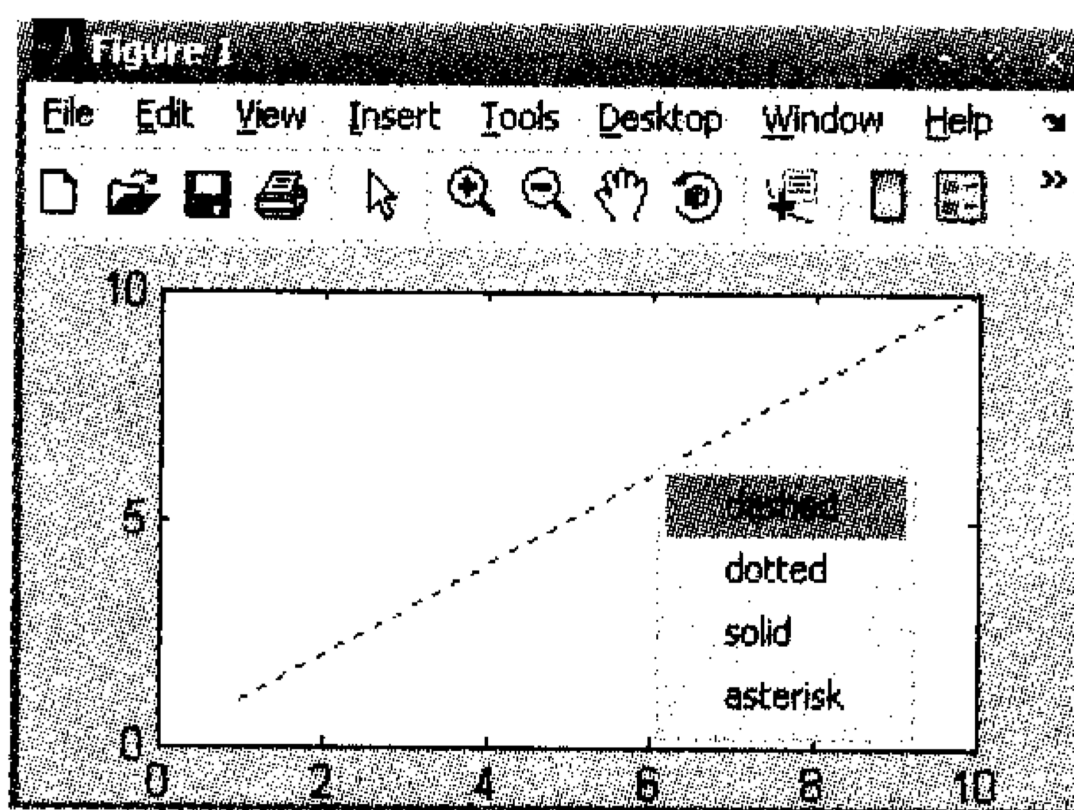


图 9-34　命令行方式创建内容式菜单

9.4.2　菜单属性

菜单属性是菜单编程中应该注意的一个重要方面， 在 MATLAB R2007 中，可以通过 get()函数获得菜单属性的属性值，通过 set()函数来设置菜单属性的属性值。当然，也可以通过前面 9.2.6 节介绍的对象属性编辑器（Property Inspector）来查看和设置菜单项的属性值，下面介绍一些常见的菜单属性。

1．Uimenu 菜单

1）Accelerator 属性

Accelerator 属性用于设置菜单的快捷键，该属性取值为一个字符。它允许用户通过一个特别的字符和一个组合键选择一个菜单选项。对于 Microsoft Windows 系统，快捷键为【Ctrl+Accelerator】，c、v 和 x 为菜单项保留字符；对于 UNIX 系统，快捷键为【Ctrl+Accelerator】，o、p、s 和 w 为菜单项保留字符，若把 Accelerator 设置成空串则移除快捷键。

2）BusyAction 属性

BusyAction 属性决定着回调程序的中断方式，取值为 Cancel 或 queue（默认）。如果回调程序正在执行，而用户在已经定义了回调程序的对象上触发了一个事件时，新事件的回调程序依据 BusyAction 的值来决定是否中断正在执行的回调程序。

- 如果 BusyAction 属性值为 cancel，事件被废除而且第二个回调函数也不会执行。

- 如果 BusyAction 属性值为 queue，且第一个回调程序的 Interruptible 属性值为“on”，则第二个回调程序将被添加到执行列队中，在第一个回调程序执行完以后开始执行。



如果中断程序是 DeleteFcn、CreateFcn 或者是一个图形对象的 CloseRequest 或 ResizeFcn 回调程序，不管对象的 Interruptible 属性是什么，它都会中断正在执行的程序。

3）Callback 属性

Callback 属性用于定义菜单对象的控制动作，取值为字符串或函数句柄。单击菜单对象，触发 callback 程序执行。拥有子菜单的菜单项在显示子菜单之前执行程序，没有子菜单的菜单项在释放鼠标的时候执行 callback 程序。

4）Checked 属性

Checked 属性用于在菜单项显示一个标志，选中的菜单项显示一个标志，如“√”，再一次单击该菜单项标志删除，Checked 属性取值为 on 或 off（默认）。



该属性对顶层菜单和父菜单没有影响。

5）Children 属性

Children 属性用于设置菜单的子菜单，取值为子菜单的句柄向量。句柄向量包含了菜单对象的所有子菜单项，可以通过该属性重排菜单顺序。

6）CreateFcn 属性

CreateFcn 属性定义一个菜单对象创建阶段执行的回调程序，取值为一个字符串或函数句柄。设置该属性对一个已经存在的菜单对象没有影响。

7）DeleteFcn 属性

DeleteFcn 属性定义删除一个菜单对象时执行的回调程序，取值为一个字符串或函数句柄。MATLAB 在销毁菜单对象的属性前执行回调程序，所以，这些属性值对回调程序仍然有用。

对于正在执行 DeleteFcn 的对象的句柄只能通过根对象 CallbackObject 属性访问，其更多的属性可以通过函数 gcbo()获得。

8）Enable 属性

Enable 属性用于设置菜单的有效和禁用，取值为 on（默认）或 off。该属性控制菜单项是否能够被选择，当属性值设置成 off 时，菜单标签为灰色，表示菜单项禁用。

9）ForegroundColor 属性

ForegroundColor 属性用于设置菜单项的标签的颜色，属性值 ColorSpec。该属性决定标签文本的颜色，默认颜色为黑色。

10）HandleVisibility 属性

HandleVisibility 属性用于控制句柄对象的访问权限，属性取值为 on（默认）、callback 或 off。该属性决定当句柄对象在父对象 children 属性的属性值是否可见。当在父对象

children 属性中句柄对象不可见时，它不能被查找对象句柄或者查询句柄属性的函数获得，这些函数通常包括 get、findobj、gca、gcf、gco、newplot、cla、clf 和 close。

- 当属性 HandleVisibility 的值为 on 时，句柄一直是可见的。
- 当 HandleVisibility 属性值为 callback 时，在回调程序或者被回调程序调用的函数中是可见的，但在命令行调用的函数中不可见，这样就有效地保护了 GUI 用户不受命令行用户的影响，同时允许回调程序完成对句柄对象的访问。
- 当 HandleVisibility 属性值为 off 时，使得句柄在所有时候都不可见，这在一个回调程序调用一个可能破坏 GUI 的函数时，可以避免对象受到不必要的影响，因此在函数执行的时候暂时隐藏它本身的句柄。

11）Interruptible 属性

Interruptible 属性决定回调程序的中断模式，其属性取值为 on（默认）或 off。如果用户触发（如单击鼠标）了一个正在执行回调程序的对象的另一个回调程序时，那么随后的回调程序试图中断先前执行的回调程序。MATLAB 根据以下因素处理回调程序。

- 正在执行回调程序的对象 Interruptible 属性值。
- 正在执行的回调程序是否包含 drawnow、figure、getframe、pause 或 waitfor 声明。
- 正在等待执行的回调程序对象的 BusyAction 属性值。

如果正在执行回调程序的对象 Interruptible 属性值为 on（默认）,那么正在执行的回调程序中断，执行下一个包含 drawnow、figure、getframe、pause 或 waitfor 之一函数的回调程序。如果对象 Interruptible 属性值为 off，那么 Busybutton 属性决定回调程序的中断方式。



如果中断回调程序是 DeleteFcn、CreateFcn 或者窗口对象的 CloseRequest 或 ResizeFcn，那么无论对象的 Interruptible 属性值是什么，都会中断当前运行的回调程序而开始执行下一个包含有 drawnow、figure、getframe、pause 或 waitfor 声明的回调程序。图形窗口对象的 WindowButtonDownFcn 属性的回调程序或者菜单对象的 ButtonDownFcn 属性的回调程序按照上述原则执行。

12）Label 属性

Label 属性用于设置菜单的标题名称，属性取值为字符串。用户可以在设置 Label 属性的字符串时使用字符“&”，用来设置菜单的快捷键。设置“&”后的那个字符出现一个下画线，用户按下【Alt】+标有下画线的字符来组合快捷键用以激活菜单。

13）Parent 属性

Parent 属性用来设置属性值为另一个父对象的句柄，取值为菜单的父对象句柄。

14）Position 属性

Position 属性用来确定菜单的位置，取值为一个标量，默认为 1。顶层菜单在菜单栏上的位置为从左至右，Position 属性为 1 的菜单表明它的位置为最左端；而下拉菜单是从上至下放置，Position 属性为 1 的菜单表明它的位置为最上端。

15）Separator 属性

Separator 属性用来设置菜单的分割条，属性值为 on 或 off（默认）。通过设置该属性在

菜单上出现一条分割线。

16）Tag 属性

Tag 属性用来标志菜单项的名字，在回调函数里面通过该名字来指定菜单项，属性值为字符串。该属性在设计交互式图形程序时，不必把对象句柄设置成全局变量或把它们作为参数传入回调函数中，直接调用该图形对象即可。用户可以定义 Tag 属性为任意字符串。

17）Type 属性

Type 属性用来标识图形对象的类，属性值为字符串（只读）。对于 Uimenu 对象，Type 的属性值一直都是“uimenu”。

18）UserData 属性

UserData 属性用来保存与菜单对象有关的信息或者数据，属性值为矩阵。用户的任何矩阵都可以和菜单对象连接，MATLAB 程序并不使用这些数据，但用户可以通过 Set()函数和 Get()函数来设置和获得它们。

19）Visible 属性

Visible 属性用来控制菜单是否可见，取值为 on（默认）或 off。默认情况时，所有的菜单都是可见的。当把它设置成 off 时，菜单不可见，但用户仍然能够查询和设置它的属性。

2．Uicontextmenu 菜单

1）BusyAction 属性

BusyAction 属性用于决定回调程序的中断方式，取值为 cancel 或 queue（默认）。如果回调程序正在执行，而用户在已经定义了回调程序的对象上触发了一个事件时，新事件的回调程序依据 BusyAction 的值来决定是否中断正在执行的回调程序。

- 如果 BusyAction 属性值为 cancel，事件被废除而且第二个回调函数也不会被执行。
- 如果 BusyAction 属性值为 queue，且第一个回调程序的 Interruptible 属性值为“on”，第二个回调程序将被添加到执行列队中，在第一个回调程序执行完以后开始执行。



如果中断程序是 DeleteFcn、CreateFcn 或者是一个图形对象的 CloseRequest 或 ResizeFcn，不管对象的 Interruptible 属性是什么，它都会中断正在执行的程序。

2）Callback 属性

Callback 属性用于定义菜单对象的控制动作，取值为字符串。但用鼠标右键单击一个对象，触发 callback 程序执行，在 uicontextmenu 菜单显示之前该程序执行，没有子菜单的菜单项在释放鼠标的时候执行 callback 程序。字符串为一个有效的 MATLAB 表达式或 M-file 的名字，表达式在 MATLAB 的命令窗口执行。

3）Children 属性

Children 属性值为矩阵，菜单项定义为内容式菜单 uicontextmenu。

4）CreateFcn 属性

CreateFcn 属性定义一个 uicontextmenu 菜单对象创建阶段执行的回调程序，取值为一个字符串或函数句柄。对 uicontextmenu 菜单对象设置的属性值必须是默认值。设置该属性对一个已经存在的 uicontextmenu 菜单对象没有影响。

5）DeleteFcn 属性

DeleteFcn 属性定义删除一个 uicontextmenu 菜单对象时执行的回调程序，取值为一个字符串或函数句柄。MATLAB 在销毁 uicontextmenu 菜单对象的属性前执行回调程序，所以，这些属性值对回调程序仍然有用。

对于正在执行 DeleteFcn 的对象的句柄只能通过根对象 CallbackObject 属性访问，其更多的属性可以通过函数 gcbo()查询。

6）HandleVisibility 属性

HandleVisibility 属性用于控制句柄对象的访问权限，属性取值为 on(默认)、callback 或 off。该属性决定当句柄对象在父对象 children 属性的属性值是否可见。当在父对象 children 属性中句柄对象不可见时，它不能被查找对象句柄或者查询句柄属性的函数获得，这些函数通常包括 get、findobj、gca、gcf、gco、newplot、cla、clf 和 close。

- 当属性 HandleVisibility 的值为 on 时，句柄一直是可见的。
- 当 HandleVisibility 属性值为 callback 时，在回调程序或者被回调程序调用的函数中是可见的，但在命令行调用的函数中不可见，这样就有效地保护了 GUI 用户不受命令行用户的影响，同时允许回调程序完成对句柄对象的访问。
- 当 HandleVisibility 属性值为 off 时，使得句柄在所有时候都不可见，这在一个回调程序调用一个可能破坏 GUI 的函数时，可以避免对象受到不必要的影响，因此在函数执行的时候暂时隐藏它本身的句柄。

用户可以设置根对象的 ShowHiddenHandles 属性值为 on 来使所有句柄对象可见，而不管它们的 HandleVisibility 属性如何设置，这对 HandleVisibility 的属性值并没有影响。

7）Interruptible 属性

Interruptible 属性决定 uicontextmenu 菜单的回调程序的中断调用模式，其属性取值为 on（默认）或 off。如果用户触发（如单击鼠标）了一个正在执行回调程序的对象的另一个回调程序时，那么随后的回调程序试图中断先前执行的回调程序。MATLAB 根据以下因素处理回调程序。

- 正在执行回调程序的对象 Interruptible 属性值。
- 正在执行的回调程序是否包含 drawnow、figure、getframe、pause 或 waitfor 声明。
- 正在等待执行的回调程序对象的 BusyAction 属性值。

如果正在执行回调程序的对象 Interruptible 属性值为 on(默认)，那么正在执行的回调程序中断，执行下一个包含 drawnow、figure、getframe、pause 或 waitfor 之一函数回调程序。如果对象 Interruptible 属性值为 off，那么 Busybutton 属性决定回调程序的中断方式。



如果中断回调程序是 DeleteFcn、CreateFcn 或者窗口对象的 CloseRequest 或 ResizeFcn，那么无论对象的 Interruptible 属性值是什么，都会中断当前运行的回调程序而开始执行下一个包含有 drawnow、figure、getframe、pause 或 waitfor 声明的回调程序。图形窗口对象的 WindowButtonDownFcn 属性的回调程序或者菜单对象的 ButtonDownFcn 属性的回调程序按照上述原则执行。

8）Parent 属性

Parent 属性用来设置属性值为另一个父对象的句柄，取值为菜单的父对象句柄。

9）Position 属性

Position 属性用于定义 Visible 的属性为 on 时的 Uicontextmenu 菜单的位置，属性取值为一个二维向量。二维向量[x y]表明 Uicontextmenu 菜单的左上角与图形窗口、面板或按钮组的左下角的水平和垂直距离。

10）Tag 属性

Tag 属性用来标志菜单项的名字，在回调函数里面通过该名字来指定菜单项，属性值为字符串。该属性在设计交互式图形程序时，不必把对象句柄设置成全局变量或把它们作为参数传入回调函数中，直接调用该图形对象即可。用户可以定义 Tag 属性为任意字符串。

11）Type 属性

Type 属性用来标识图形对象的类，属性值为字符串（只读）。对于 Uicontextmenu 对象，Type 的属性值一直都是“Uicontextmenu”。

12）UserData 属性

UserData 属性用来保存与菜单对象有关的信息或者数据，属性值为矩阵。用户的任何矩阵都可以和 uicontextmenu 菜单对象连接，MATLAB 程序并不使用这些数据，但是用户可以通过 Set()函数和 Get()函数来设置和获得它们。

13）Visible 属性

Visible 属性用来控制 Uicontextmenu 菜单是否可见，取值为 on（默认）或 off。默认情况时，所有的菜单都是可见的。把 Visible 属性值设置成 off 时，菜单不可见，但用户仍然能够查询和设置它的属性。

9.5 用户控件

在图形界面程序中，最常见的对象除了前面介绍的菜单之外就是控件了。控件作为另一种图形对象和菜单对象一起构建起图形用户界面。用户通过菜单和控件可以创建功能强大、界面美观的图形用户界面。

9.5.1 MATLAB 控件介绍

MATLAB R2007 提供了许多控件，用户可通过鼠标或键盘激活它们。MATLAB R2007 支持的控件类型有 10 种，它们是复选框 Check boxes、可编辑文本框 Editable text fields、框架 Frames、列表框 List boxes、弹出菜单 Pop-up menus、命令按钮 Push buttons、单选按钮 Radio buttons、滑动条 Sliders、静态文本标签 Static text Labels 和开关按钮 Toggle buttons，下面分别对它们进行介绍。

1）复选框 Check boxes

复选框为一个带有静态说明文本的选择小框，一般在用户自由选择多个候选项时使用。用鼠标单击复选项或先用【Tab】键转移焦点至复选项，再按空格键激活复选框。选中时，

复选框的 Value 属性值为 1；没有选中时，Value 属性值为 0。

2）可编辑文本框 Editable text fields

用户可通过可编辑文本框输入或修改文本信息，然后把信息存储在文本框的 string 属性里。如果文本框中属性 Max-Min>1，则文本框是多行模式。对于多行的文本编辑框，用户可以按键盘的方向箭头滑动垂直滑动条。

3）框架 Frames

框架是一个为图形窗口提供视觉上封闭区域的方框。它能够使得有着群组关系的控件的用户界面更容易被理解。框架是没有回调程序的，只有其他控件出现框架中。

框架是不透明的，所以，用户设置框架和框架中控件的顺序很重要，一般是框架要先于控件设置，否则，框架将会覆盖原来定义好的控件。

4）列表框 List boxes

列表框显示一个选项列表，使用户能够选择一个或者多个选项。列表框的 Min 和 Max 属性控制着选择模式。

- 当 Max-Min>1 时，允许选择多个选项。
- 当 Max-Min<=1 时，只能选择单个选项。

属性 Value 指示的是列表项的索引值，当单击列表项的鼠标松开以后将调用一个回调程序。在列表框中，单击和双击鼠标的触发结果是不同的。

5）弹出菜单 Pop-up menus

当用户单击信息框时，显示一个选择列表。当弹出菜单没有打开的时候，信息框显示的是当前的选择项。弹出菜单对于用户在选择大量而互不相同的选择的时候是非常有用的，若不使用该控件一般选择设置一系列的单选按钮。

6）命令按钮 Push buttons

命令按钮是最常见的控件之一。按下按钮将产生一个行为，用鼠标单击按钮或者获得焦点以后按【Enter】键激活按钮。

7）单选按钮 Radio buttons

单选按钮类似复选框，但在一组单选按钮中一次只能选中一个，因为每个按钮项之间是互斥的。用鼠标单击按钮或移动箭头激活按钮，选中时 Radio buttons 的 Value 属性值为 1，没有选中时 Value 属性值为 0。

8）滑动条 Sliders

滑动条由 3 部分组成，即滑动指示条、滑动槽和滑动槽两端的箭头。可以通过拖动滑动条或者单击滑动槽两端的箭头来改变滑动条的值。用户可以设置滑动指示条的 minimum 属性值、maximum 属性值和当前的 Values 属性值。

9）静态文本标签 Static text Labels

静态文本标签用来显示一行文本信息。静态文本常常用来显示其他控件的相关信息，给用户提供指导或显示滑动条相关的值。用户不能够改变它的值，同时也没有回调程序。

10）开关按钮 Toggle buttons

当用鼠标单击并显示其状态时，开关按钮控制着回调程序的执行，它在创建工具栏时是非常有用的。

9.5.2　控件的创建

MATLAB R2007 中，创建控件也像前面创建菜单一样可以通过 GUI 设计工具和命令行两种方式。

1. GUI 设计工具

如前面 9.2 节中所介绍的，打开对象编辑器（Layout Editor），然后在窗口左边的控件栏中选择需要的控件添加到控件对象设计区。并且还通过对象属性编辑器查看和设置各个控件对象的属性值。具体的使用请参考 9.2 节图形用户界面设计工具相关内容。下面是运用 GUI 设计工具中控件对象创建的图形用户界面程序，如图 9-35 所示。

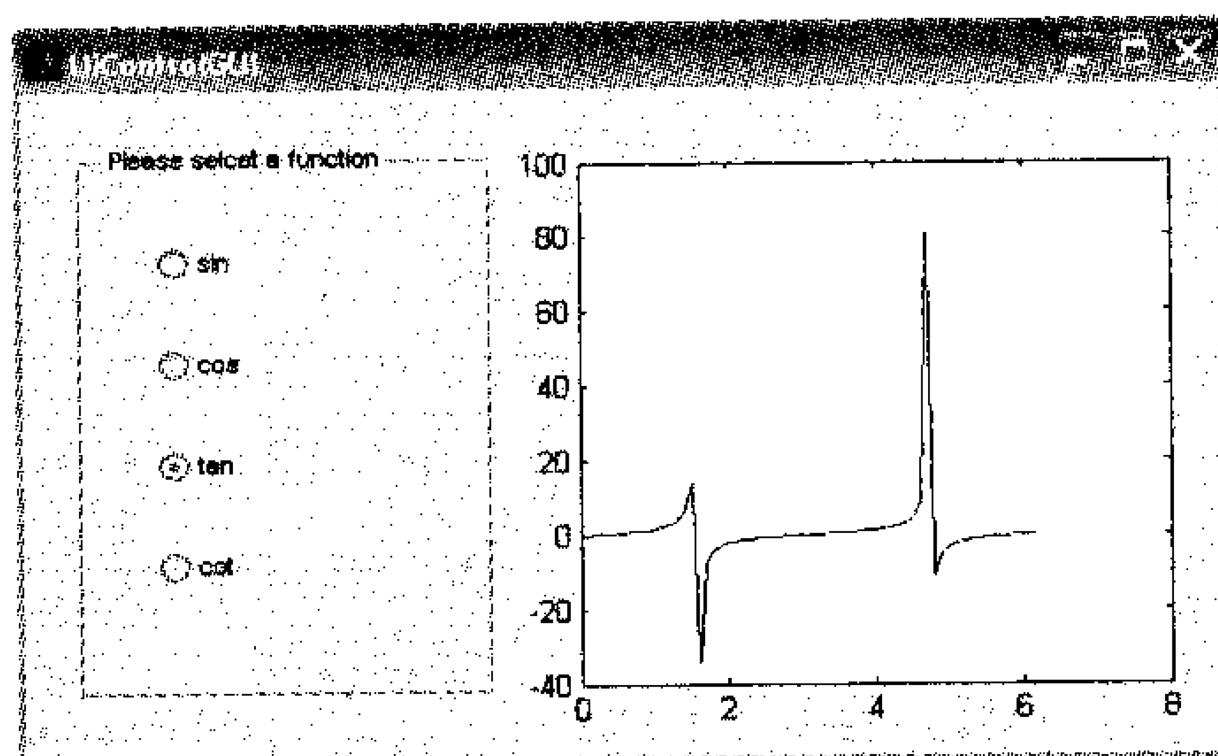


图 9-35　GUI 设计工具设计的图形界面程序

2. 命令行创建方式

MATLAB R2007 提供了函数 Uicontrol()来创建用户界面的控件对象。

函数格式如下：

```
handle = uicontrol('PropertyName',PropertyValue,...) %创建控件对象并指定相关的属性值，参数PropertyName 为控件属性名，PropertyValue 为属性名对应的属性值
handle = uicontrol(parent,'PropertyName',PropertyValue,...)
%参数 parent 为控件对象所在的父对象即图形窗口的句柄值
handle = uicontrol                %在当前图形对象创建一个命令按钮，所有属性为默认值
uicontrol(uich)
 %由句柄参数 uich 指定控件的焦点，当控件对象被选择后就执行预先确定的行为
```

例 9.16　创建一个命令按钮。

```
>> h1=uicontrol('style','pushbutton','string','Open');
```

运行结果如图 9-36 所示。

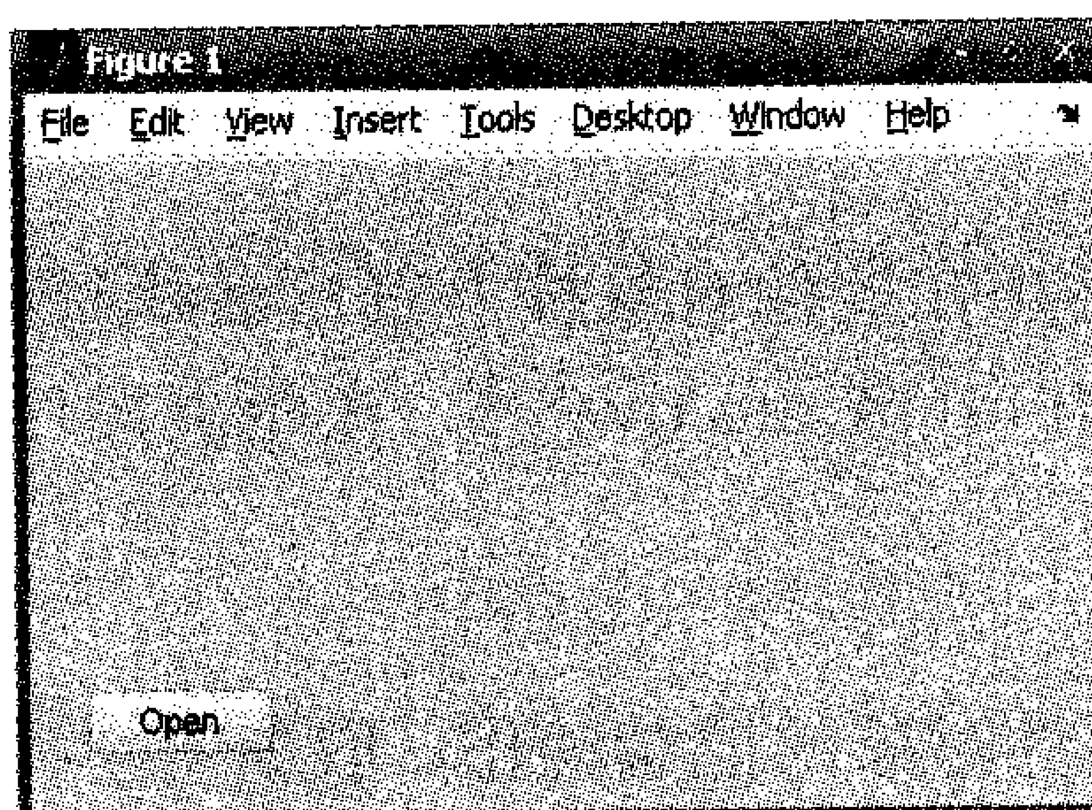


图 9-36　命令行方式创建控件

9.5.3 控件的属性及设置

1．控件的属性

控件对象的属性主要用来描述各种控件的颜色、大小、位置、字体、显示信息、所属类型，以及回调程序的控制情况等信息。下面将介绍控件对象的一些常见属性。

1）BackgroundColor 属性

BackgroundColor 属性用于定义控件的背景颜色，可以用一组向量或者 MATALB 预定义的 8 种颜色名字来设置属性值。向量组采用的是 RGB 颜色，里面的元素取值为[0 1]内的数值，如[1 1 0]为黄色。常见的 RGB 颜色值和颜色名如表 9-3 所示。

表 9-3　RGB　颜色值和颜色名

RGB value	Short Name	Long Name
[1 1 0]	y	yellow
[1 0 1]	m	magenta
[0 1 1]	c	cyan
[1 0 0]	r	red
[0 1 0]	g	green
[0 0 1]	b	blue
[1 1 1]	w	white
[0 0 0]	k	black

2）BusyAction

BusyAction 属性用于决定回调程序的中断方式，取值为 Cancel 或 queue（默认）。如果回调程序正在执行，而用户在已经定义了回调程序的对象上触发了一个事件时，新事件的回调程序依据 BusyAction 的值来决定是否中断正在执行的回调程序。

- 如果 BusyAction 属性值为 cancel，事件被废除而且第二个回调函数也不会被执行。
- 如果 BusyAction 属性值为 queue，且第一个回调程序的 Interruptible 属性值为“on”，第二个回调程序将被添加到执行列队中，在第一个回调程序执行完以后开始执行。



如果中断程序是 DeleteFcn 、CreateFcn 回调程序或者是一个图形对象的 CloseRequest 或 ResizeFcn 回调程序，不管对象的 Interruptible 属性是什么，它都会中断正在执行的程序。

3）ButtondownFcn 属性

ButtondownFcn 属性定义当在控件对象按下鼠标后执行的回调程序，属性取值为字符串。当控件的 Enable 属性值为 off 或 inactive 时，在距离控件 5 像素的范围内按下鼠标时，仍然会激活回调程序。

4）Callback 属性

Callback 属性用于定义控件对象的控制动作，取值为字符串。当激活一个控件对象时，就触发 callback 程序执行。字符串为一个有效的 MATLAB 表达式或 M-file 的名字。表达式

在 MATLAB 的命令窗口执行。静态文本标签没有相关的回调程序，对于可编辑的文本框控件，在输入需要的内容后可以按以下键激活 callback 属性。

- 单击另一个组件、菜单条或者图形用户界面的背景。
- 对于单行的文本编辑框，按【Enter】键。
- 对于多行的文本编辑框，按【Ctrl+Enter】组合键。

5）Cdata 属性

Cdata 属性定义在控件对象上显示的图像的颜色值，取值为矩阵。该矩阵一般为一个 1 行 3 列的向量，里面的元素取值为[0 1]内的数值。

6）Children 属性

Children 属性取值为一个矩阵，在控件对象中没有 Children，所以矩阵为空。

7）Clipping 属性

Clipping 属性对控件对象没有影响，取值为 on（默认）或 off。

8）CreateFcn 属性

CreateFcn 属性定义一个控件对象创建阶段执行的回调程序，取值为一个字符串或函数句柄。对控件对象设置的属性值必须是默认值。MATLAB 在 CreateFcn 回调程序执行前设置了所有控件的属性值，所以，这些属性值对于回调程序是有用的。设置该属性对一个已经存在的控件对象没有影响。用户可以为所有新创建的控件定义默认的 CreateFcn 回调程序。

9）DeleteFcn 属性

DeleteFcn 属性定义删除一个控件对象时执行的回调程序，取值为一个字符串或函数句柄。MATLAB 在销毁控件对象的属性前执行回调程序，所以这些属性值对回调程序仍然有用。

对于正在执行 DeleteFcn 的对象的句柄只能通过根对象 CallbackObject 属性访问，其更多的属性可以通过函数 gcbo()查询。

10）Enable 属性

Enable 属性定义控件的有效和失效状态，取值为 on（默认）、inactive 或 off。当属性为 on 时，控件是可操作的；为 inactive 时，控件不可操作，但控件外观没有变；当属性为 off 时，控件不可用，同时控件为灰色。

- 当 Enable 属性值为 on 时，用鼠标左键单击控件对象，MATLAB 依次执行以下几个行为。
  - 设置图形窗口的 SelectionType 属性。
  - 执行控件的 ClickedCallback 程序。
  - 不设置图形窗口的 CurrentPoint 属性，也不执行控件的 ButtonDownFcn 程序和图形窗口的 WindowButtonDownFcn 回调程序。
- 当 Enable 属性值为 off 时，用鼠标左键单击控件对象或属性值为任意时，用鼠标右键单击控件，MATLAB 依次执行以下几个行为。
  - 设置图形窗口的 SelectionType 属性。
  - 设置图形窗口的 CurrentPoint 属性。
  - 执行图形窗口的 WindowButtonDownFcn 回调程序。

11）Extent 属性

Extent 属性定义了控件的字符串的大小，属性取值为四元素的向量，[0,0, width,height]。第一、二个元素常常为 0，width 和 height 表示长方形的尺度，所有的单位由 Units 属性指定。

12）FontAngle属性

FontAngle属性定义了字体的倾向，取值为 normal(默认)、italic 或 oblique。MATLAB 使用该属性从用户特定的系统选择一个字体。当设置该属性为 italic 或 oblique 时，如果系统的字库中有这种字体，则从系统中选择一种倾斜字体。

13）FontName 属性

FontName 属性定义字体的名字，显示字体的类型，取值为字符串。为了显示和打印正常，字体必须是计算机操作系统支持的字体，默认的属性值为系统的字体。用户可以根据自己的使用环境，通过设置根对象的 FixedWidthFontName 属性值使 MATLAB 采用特定的字体。

14）FontSize 属性

FontSize 属性定义了字体的大小，取值为一个数值。字体的大小单位由 FontUnits 属性决定，默认的属性值为系统默认的字体大小。

15）FontUnites 属性

FontUnites 属性用于设置字体大小的单位，取值为 points(默认)、normalized、inches、pixels 或 Centimeters。该属性值的设置影响 Fontsize 属性，Normalized 设置 Fontsize 属性的属性值作为控件对象大小的一部分。

16）FontWeight 属性

FontWeight 属性用于设置字体的粗细程度，取值为 light、normal(默认值)、demi 或 bold。

17）ForegroundColor 属性

ForegroundColor 属性定义了控件上显示的文本的颜色，颜色取值使用一个三元素的 RGB 向量或者预先定义好的颜色名，默认为黑色。

18）HandleVisibility 属性

HandleVisibility 属性用于控制句柄对象的访问权限，属性取值为 on(默认)、callback 或 off。该属性决定当句柄对象在父对象 children 属性的属性值是否可见。当在父对象 children 属性中句柄对象不可见时，它不能被查找对象句柄或者查询句柄属性的函数获得，这些函数通常包括 get、findobj、gca、gcf、gco、newplot、cla、clf 和 close。

- 当属性 HandleVisibility 的值为 on 时，句柄一直是可见的。
- 当 HandleVisibility 属性值为 callback 时，在回调程序或者被回调程序调用的函数中是可见的，但在命令行调用的函数中不可见，这样就有效地保护了 GUI 用户不受命令行用户的影响，同时允许回调程序完成对句柄对象的访问。
- 当 HandleVisibility 属性值为 off 时，使得句柄在所有时候都不可见，这在一个回调程序调用一个可能破坏 GUI 的函数时，可以避免对象受到不必要的影响，因此在函数执行的时候暂时隐藏它本身的句柄。

用户可以设置根对象的 ShowHiddenHandles 属性，使所有的句柄不管它们的 HandleVisibility 属性如何设置都可见，这个设置不会影响 HandleVisibility 的属性。

19）HitTest 属性

HitTest 属性定义是否可以被鼠标单击，取值为 on 或 off，HitTest 属性对控件对象没有影响。

20）HorizontalAlignment 属性

HorizontalAlignment 属性用于定义控件上显示字符的对齐方式，属性取值为 left、center（默认）或 right。

- left：显示的字符居左对齐。
- center：显示的字符居中对齐。
- right：显示的字符居右对齐。

在 Microsoft Windows 系统中，该属性只对控件对象的可编辑文本框和静态文本标签有影响。

21）Interruptible 属性

Interruptible 属性决定回调程序的中断模式，其属性取值为 on（默认）或 off。如果用户触发（比如单击鼠标）了一个正在执行回调程序的对象的另一个回调程序时，那么随后的回调程序试图中断先前执行的回调程序。MATLAB 根据以下因素处理回调程序。

- 正在执行回调程序的对象的 Interruptible 属性值。
- 正在执行的回调程序是否包含 drawnow、figure、getframe、pause 或者 waitfor 声明。
- 正在等待执行的回调程序对象的 BusyAction 属性值。

如果正在执行回调程序的对象 Interruptible 属性值为 on(默认),那么正在执行的回调程序中断，执行下一个包含 drawnow、figure、getframe、pause 或 waitfor 之一函数的回调程序。如果对象 Interruptible 属性值为 off，那么 Busybutton 属性决定回调程序的中断方式。

注意　如果中断回调程序是 DeleteFcn、CreateFcn 或者窗口对象的 CloseRequest 或 ResizeFcn，那么无论对象的 Interruptible 属性值是什么，都会中断当前运行的回调程序而开始执行下一个包含有 drawnow、figure、getframe、pause 或 waitfor 声明的回调程序。图形窗口对象的 WindowButtonDownFcn 属性的回调程序或者控件对象的 ButtonDownFcn 属性的回调程序按照上述原则执行。

22）KeyPressFcn 属性

KeyPressFcn 属性用来定义按下键盘键时触发一个回调函数，取值为字符串或者句柄函数。当一个回调程序的控件对象获得焦点时，按下一个键将触发一个回调程序。如果没有控件对象获得焦点，图形对象的按键回调程序将触发。KeyPressFcn 的属性值可以是一个函数的句柄、一个 M-file 的文件名或者是任意一个合法的 MATLAB 表达式。

- 如果指定的属性值为 M-file 的文件名，回调程序可以查询图形对象的 CurrentCharacter 属性来确定哪个按键被按下以及哪些按键被限制触发回调程序。
- 如果指定的属性值为一个函数句柄，则回调程序能够从按键事件的数据结构体参数中检索到它的相关信息。

23）ListboxTop 属性

ListboxTop 属性只应用于列表框控件，用来显示列表框最顶端的字符串的索引号，取值为一个标量。ListboxTop 属性是 String 属性定义字符串向量的一个元素的索引值，而且取值必须在 1 和字符串向量的长度之间。

24）Max 属性

Max 属性定义的是 Value 值允许的最大值，参数取值为一个标量。不同类型的控件对象阐述的 Max 含义都不相同。

- 在复选框中，复选框被选中时，Value 的属性值为 Max。
- 在可编辑文本框中，如果 Max-Min>1，则可编辑文本框接受多行输入；如果 Max-Min<=1，则可编辑文本框只接受单行输入。
- 在列表框中，如果 Max-Min>1，则列表框允许选择多个列表项；如果 Max-Min<=1，则列表框只能选择单个列表项。
- 在单选按钮中，当单选按钮被选中时，Value 的属性值为 Max。
- 在滑动条中，定义了滑动条的最大值，且必须比最小值大，默认值为 1。
- 在开关按钮中，当关按钮被选中时，Value 的属性值为 Max，默认值为 1。
- 弹出菜单、命令按钮和静态文本标签没有 Max 属性。

25）Min 属性

Min 属性定义的是 Value 值允许的最小值，参数取值为一个标量。不同类型的控件对象阐述的 Min 含义都不相同。

- 在复选框中，当复选框没有被选中时，Value 的属性值为 Min。
- 在可编辑文本框中，如果 Max-Min>1，则可编辑文本框接受多行输入；如果 Max-Min<=1，则可编辑文本框只接受单行输入。
- 在列表框中，如果 Max-Min>1，则列表框允许选择多个列表项；如果 Max-Min<=1，则列表框只能选择单个列表项。
- 在单选按钮中，当单选按钮没有被选中时，Value 的属性值为 Min。
- 在滑动条中，定义了滑动条的最小值，且必须比最大值小，默认值为 0。
- 在开关按钮中，当开关按钮没有被选中时，Value 的属性值为 Min，默认值为 0。
- 弹出菜单、命令按钮和静态文本标签没有 Min 属性。

26）Parent 属性

Parent 属性定义的是控件对象的父对象，一般为图形窗口，取值为父对象的句柄。用户通过设置 Parent 属性值为另一父对象的句柄，可以把本控件移到另一个图形窗口、面板或按钮群组对象。

27）Position 属性

Position 属性决定控件的位置和大小，取值为[left bottom width height]。其中 left 和 bottom 表示控件对象的左下角与父对象的左下角的距离，而 width 和 height 表示控件对象的宽度和高度，所有的长度单位由 Units 属性指定。

28）Selected 属性

Selected 属性决定控件对象是否被选中，取值为 on（默认）或 off。当 Selected 属性为 on，且 SelectionHighlight 属性也为 on 时，MATLAB 显示控件对象的句柄。例如，可以通

过定义 ButtonDownFcn 属性来定义这个属性，允许用户用鼠标选择控件对象。

29）SelectionHighlight 属性

SelectionHighlight 属性用于设置控件被选中时是否显示其状态，取值为 on（默认）或 off（只读）。SelectionHighlight 属性可以与 Selcet 属性一起使用。当 SelectionHighlight 属性值为 off 时，MATLAB 得不到句柄。

30）SliderStep 属性

SliderStep 属性定义滑动条每次滑动的步长，属性取值为二元向量[min_step max_step]，min_step 表示最小步长，max_step 表示最大步长。元素值必须是[0 1]之间的数值，且默认值为[0.01 0.10]。当用鼠标单击两端的箭头时，每次滑动 1%；当单击滑动槽时，每次滑动 10%。

31）String 属性

String 属性用来显示在控件上的文本，取值为字符串。对于复选框、可编辑文本框、命令按钮、单选按钮、静态文本标签、开关按钮文本显示在控件对象上；对于列表框和弹出菜单文本显示在列表项中。

32）Style 属性

Style 属性决定了创建控件对象的类型，取值可以为 pushbutton（默认）、togglebutton、radiobutton、checkbox、edit、text、slider、frame、listbox 和 popupmenu 等。

33）Tag 属性

Tag 属性用来标志控件的名字，在回调函数里面通过该名字来指定控件对象，属性值为字符串。该属性在设计交互式图形程序时，不必把对象句柄设置成全局变量或把它们作为参数传入回调函数中，直接调用该图形对象即可。用户可以定义 Tag 属性为任意字符串。

34）TooltipString 属性

TooltipString 属性用以简单说明控件对象，取值为字符串。当用户移动鼠标指针到控件上时，显示该属性值。

35）Type 属性

Type 属性用来标识图形对象的类，属性值为字符串（只读）。对于控件对象，Type 的属性值一直都是“uicontrol”。

36）UIContextMenu 属性

UIContextMenu 属性使一个文本菜单 contextmenu 对象与控件相关联，属性值为对象句柄。当用鼠标右键单击控件对象的时候会显示 contextmenu 菜单，用户可以通过 uicontextmenu()函数创建一个 contextmenu 菜单。

37）Units 属性

MATLAB 用 Units 属性来解释 Extent 属性和 Position 属性，值可以为 pixels（默认）normalized（GUIDE 中默认值）、inches、centimeters、points 和 characters 等。

38）UserData 属性

UserData 属性用来保存与控件对象有关的信息或者数据，属性值为矩阵。MATLAB 并不使用这些数据，但是用户可以通过 Set()函数和 Get()函数来设置和获得它们。

39）Value 属性

Value 属性用于设置控件的当前值，取值为标量或矢量。不同类型的控件值是不同的。

- 在复选框中，被选中时，该属性值为 Max；没被选中时，该属性值为 Min。
- 在列表框中，该属性为向量形式，表示选中了多个列表项，为 1 时为列表框的第一项。
- 在弹出菜单中，该属性表示被选中的菜单项的索引值，1 为菜单列表的第一项。
- 在单选按钮中，被选中时，该属性值为 Max；没被选中时，该属性值为 Min。
- 在滑动条中，该属性表示滑动条指示的当前值。
- 在开关按钮中，被选中时，该属性值为 Max；没被选中时，该属性值为 Min。
- 在可编辑文本框中，命令按钮和静态文本标签是没有 Value 属性的。

用户可以通过鼠标来设置 Value 的属性值，也可以通过 Set()函数来设置它。

40）Visible 属性

Visible 属性用来控制控件是否可见，取值为 on（默认）或 off。默认情况时，所有的控件对象都是可见的。把它设置成 off 时，控件不可见，但用户仍然能够查询和设置它的属性。

2．控件属性设置

控件对象的属性设置也有两种方式：一种是通过 GUI 设计工具，另一种是命令行方式。在 GUI 设计工具中，设置控件的属性非常简单、方便，双击对象或先选择对象然后单击工具栏中对象属性编辑器（Property Inspector），就可以在出现的编辑器中设置对象的属性，具体设置可参考 9.2.6 节对象属性编辑器相关内容。在命令行方式中，用函数 Uicontrol 创建一个控件的同时就可以设置控件的属性值。也可以分别利用函数 get()和 set()来获得和设置属性的属性值。

下面举例介绍一下命令行方式。

f9_3.m

```
>>whitebg('w') %create a figure with a white color scheme
set(0,'DefaultAxesColorOrder',[0 0 0],...
        'DefaultAxesLineStyleOrder','-|--|:|-.');
>>Z=peaks; plot(1:49,Z(4:7,:));
```

设置图形对象属性值为默认时，获得的结果如图 9-37 所示。

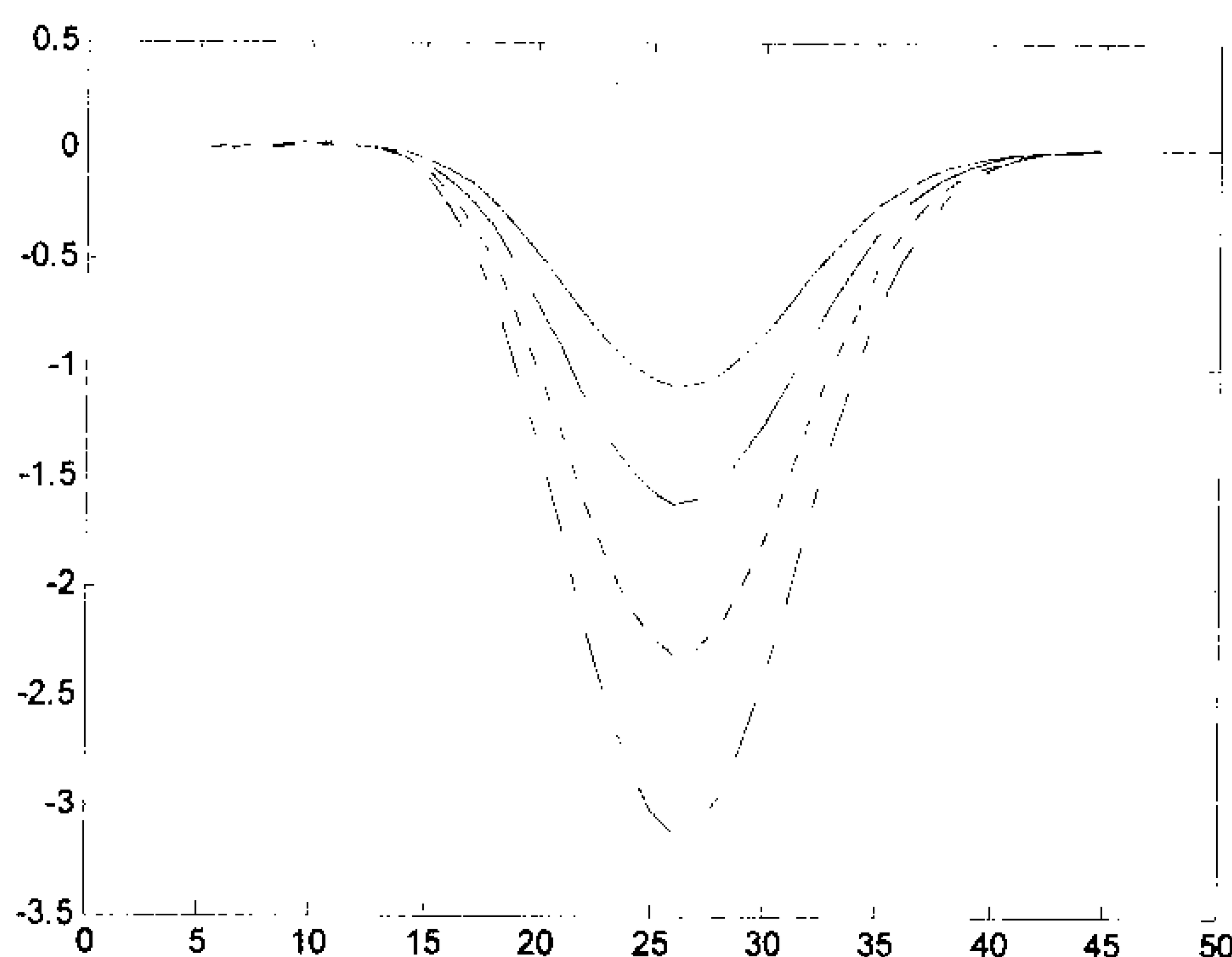


图 9-37　设置对象为默认属性值

```
set(0,'DefaultAxesColorOrder',[1 0 0],...
```

```
'DefaultAxesLineStyleOrder','-|--|:|-.'); %设置对象新的属性值
>>Z=peaks; plot(1:49,Z(4:7,:));
```

重新设置图形对象属性值为红色，得到的结果如图 9-38 所示。

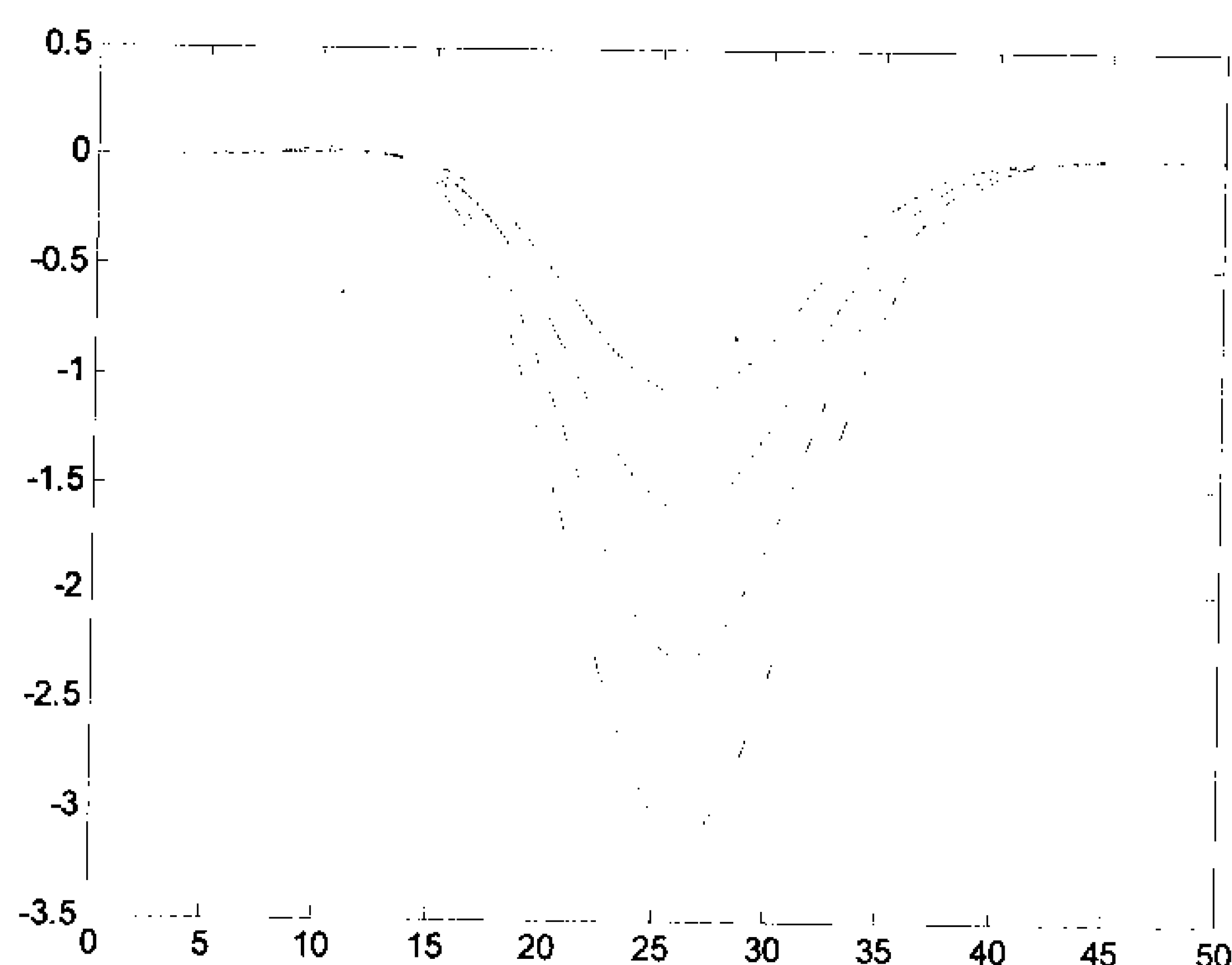


图 9-38　设置对象为新的属性值

f9_4.m

```
>> [x,y,z]=peaks;
[c,h]=contour(x,y,z);
set(h,'linewidth',1,'Linestyle',':','Color','g');
```

创建 Peaks()函数的等值线图，并且设置等值线的形状，宽度和颜色等显示结果如图 9-39 所示。

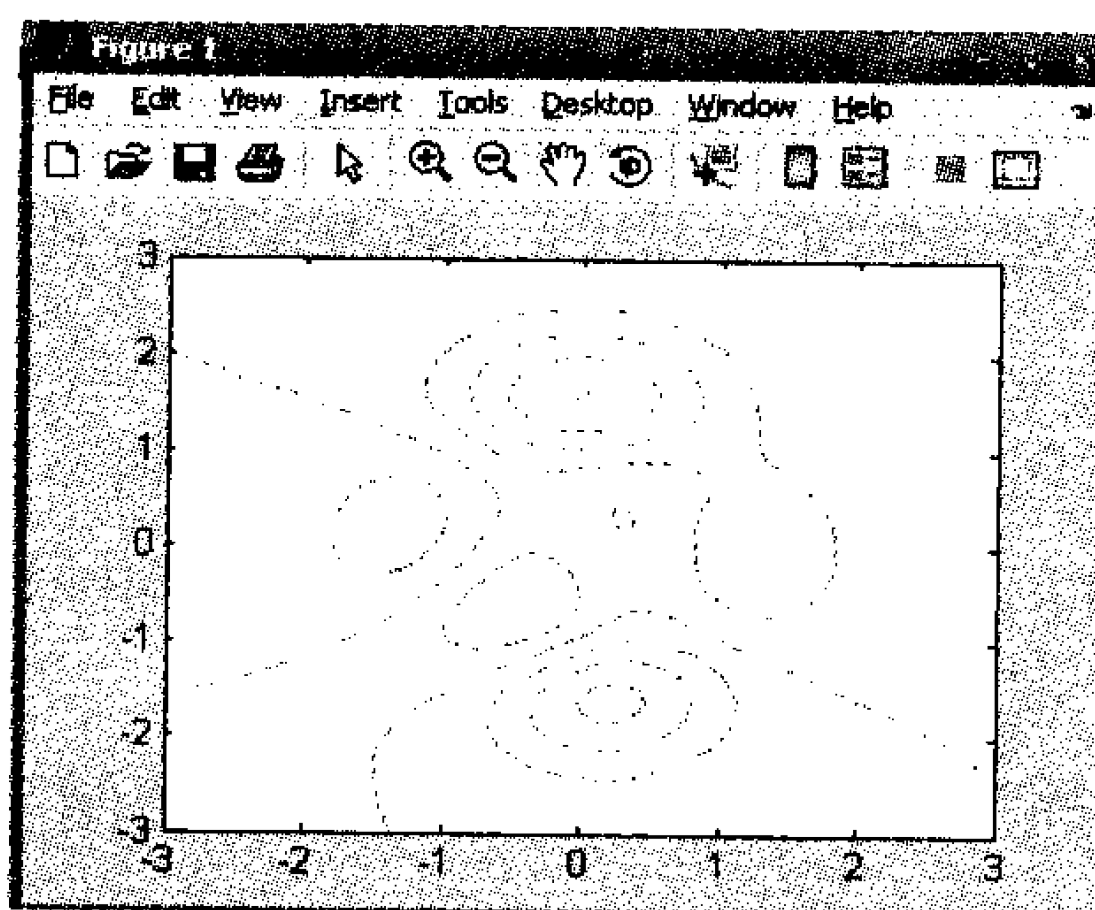


图 9-39　设置图形对象属性

f9_5.m

```
>> h=uicontrol('style','pushbutton',...
'Units','normalized',...
'Position',[.5 .5 .2 .1],...
'String','Try','callback','set(h,''Position'',[.5*rand .7*rand .3 .2])');
```

该程序在图形窗口创建一个按钮，开始给它位置赋一个初值，当用鼠标单击按钮的时候，按钮的位置通过程序 set()函数设置而不断改变，显示结果如图 9-40 所示。

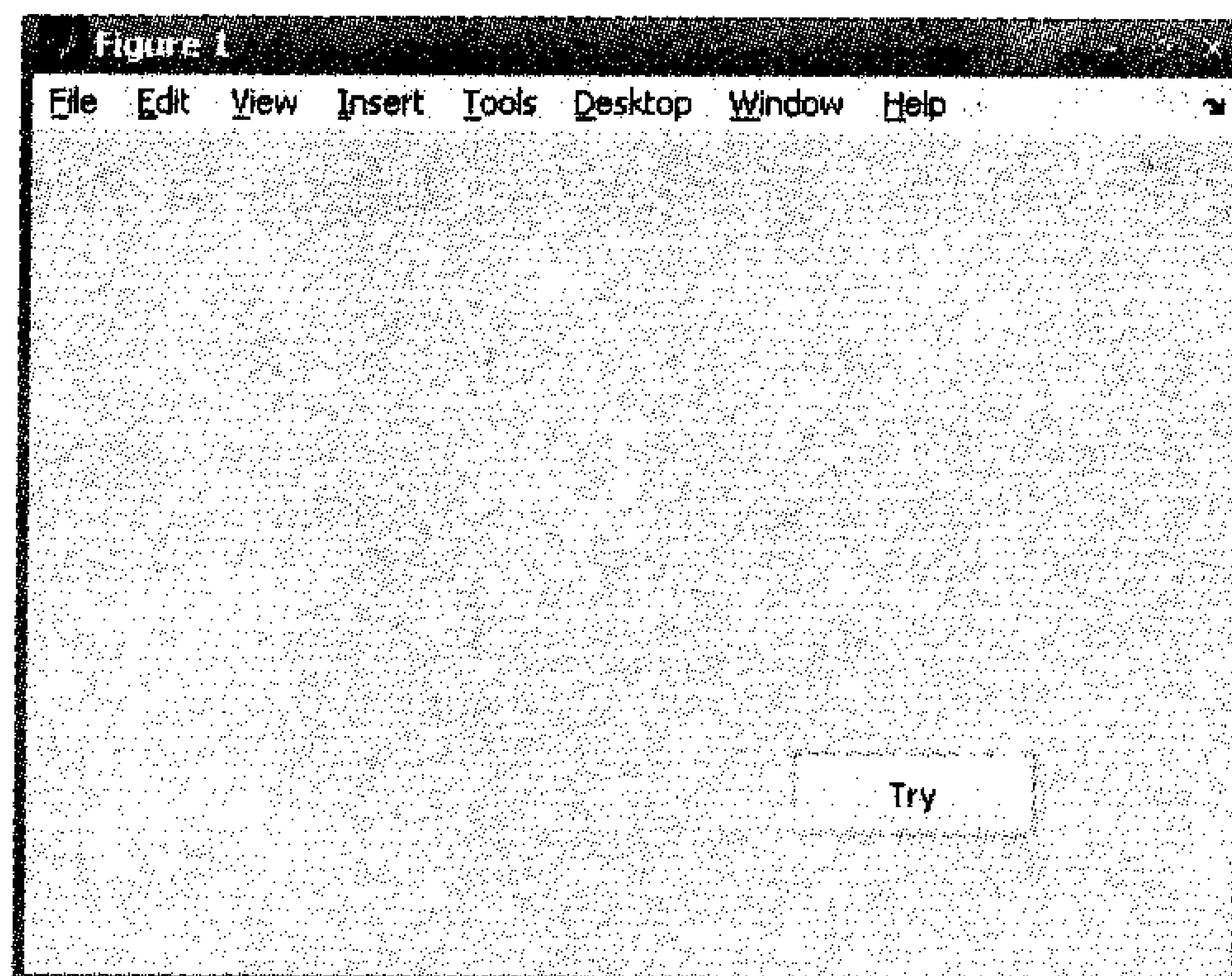


图 9-40　改变命令按钮的位置属性

9.5.4　鼠标操作

鼠标作为计算机最常见的输入设备，在图形界面程序的操作中发挥了非常重要的作用。许多面向对象的程序设计语言都纷纷为其定义了鼠标操作事件，比如，鼠标单击或者双击，释放鼠标和鼠标移动等动作都会产生一个事件。在 MATLAB R2007 中，也提供了对图形用户界面程序鼠标操作的支持。鼠标操作一般激活菜单 callback 属性和图形窗口对象与控件对象的 WindowsButtondownFcn 属性、WindowsButtonupFcn 属性、WindowsButtonmotionFcn 属性或 callback 属性。下面介绍以下几个鼠标操作。

1）按下鼠标

不同的鼠标操作会激活不同的回调程序，鼠标位于不同位置的操作也会有不同的响应。

- 当鼠标指针位于图形对象的正上方时，单击鼠标，图形对象变成当前的激活对象，这时将激活图形对象 Callback 属性中定义的属性值。
- 当鼠标指针不在图形对象的正上方，但位于离对象 5 像素时，图形对象变成当前的激活对象，图形窗口对象的 WindowsButtondownFcn 属性被激活，随后图形对象的 ButtondownFcn 属性也被激活。
- 当鼠标指针位于图形窗口对象的正上方时，那么图形窗口对象变成当前的激活对象，窗口对象的 WindowsButtondownFcn 属性被激活，同时还更新窗口属性 CurrentObject 为选中对象句柄值、更新属性 CurrentPoint 为当前鼠标指针的位置坐标值、更新窗口属性 Selectiontype 的属性值。

2）移动鼠标

当鼠标指针位于图形窗口对象内时，移动鼠标，那么图形窗口对象的 WindowsButtonmotionFcn 属性被激活，同时更新属性 CurrentPoint 为当前鼠标指针的位置坐标值。但是，如果 WindowsButtonmotionFcn 属性没有被定义，那么鼠标的动作不会激活任何属性，同时也不会更新任何新的属性。

3）释放鼠标

当在图形窗口对象中释放鼠标时，MATLAB 将会更新当前图形窗口属性 CurrentPoint

的属性值，同时激活 WindowsButtonupFcn 属性，同样，如果 WindowsButtonupFcn 属性没有被定义的时候，释放鼠标什么属性值也不会更新，任何属性也不会激活。

9.6 图形用户界面编程

图形用户界面程序设计包括界面外观设计和回调程序的编写，运用 GUIDE 可以非常方便地创建好用户界面，但是图形界面程序要实现的功能主要还是靠回调程序来完成的，图形编程就显得非常重要。前面例子较多介绍的是命令行形式图形用户界面编程，在数据比较多和程序比较大的情况下，这种方法将在很大程度上影响效率。这里将介绍几种常用的方法来编写图形用户界面程序。

9.6.1 全局变量

全局变量是在函数的公共区定义的，根据 MATLAB 的命名规则，一般全局变量名为大写。在代码比较简单的程序中，全局变量的使用会使得程序代码更加简洁。下面举例介绍全局变量在 GUI 设计中的使用。

本例在创建的图形窗口中绘出两个子图，然后通过全局变量分别对各个子图的显示属性进行设置。

f9_6.m

```
function globalGUI()
global    AH1 AH2                    %定义全局变量 AH1,AH2,AH3,AH4
t=0:0.1:2*pi;
s=sin(t);
c=cos(t);
fh=figure('defaultAxesColor',[.6 .6 .6]); %设置坐标轴的初始颜色值

%--------------------依次设置第一个坐标轴的线型、坐标轴的线宽、坐标轴的前景颜色
--------------------
AH1=subplot(2,2,1);
grid on
set(AH1,'DefaultLineLineStyle','-');
set(AH1,'LineWidth',1);
set(AH1,'Color',[.9 .9 .9]);
line('Xdata',t,'Ydata',s)

%-----------同第一个坐标轴 ------------------
AH2=subplot(2,2,2);
grid on
set(AH2,'DefaultLineLineStyle','-.');
set(AH2,'LineWidth',2);
set(AH2,'Color',[.8 .8 .8]);
line('Xdata',t,'Ydata',c)
```

执行该函数显示结果如图 9-41 所示。

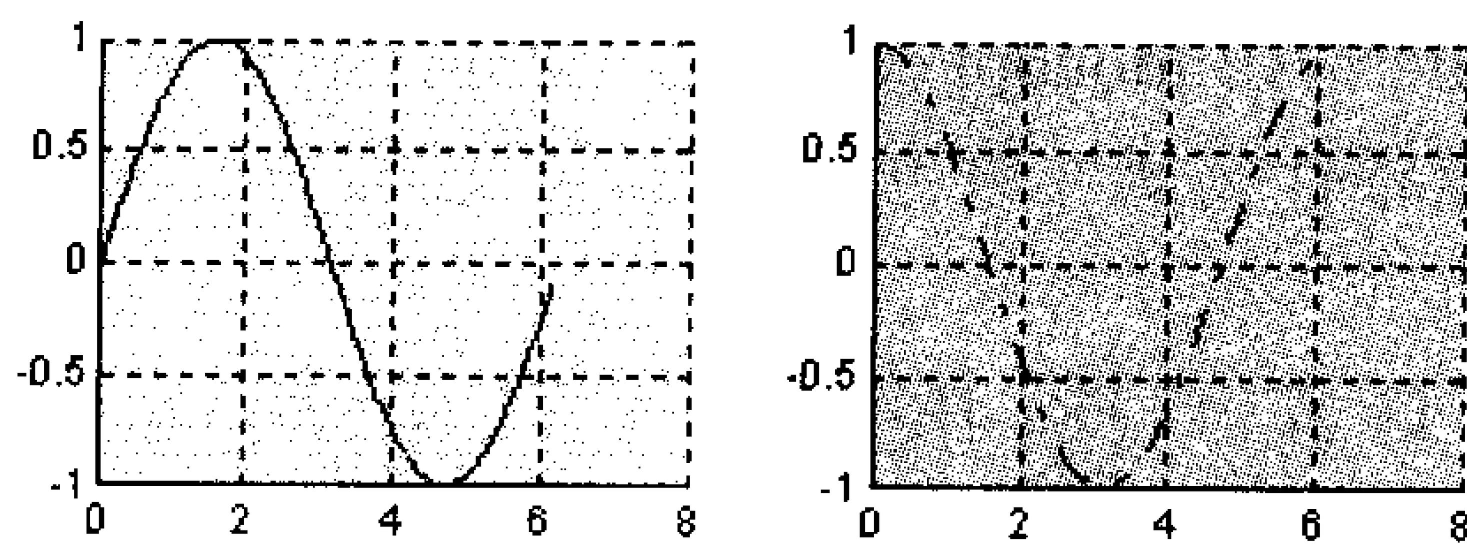


图 9-41　使用全局变量的函数显示的图形

9.6.2　用户数据属性（UserData）

在代码比较简单的程序中，全局变量能够使得整个程序代码清晰、简洁。但是，在更加复杂的 GUI 程序中，要尽可能避免使用全局变量。因为，此时它很容易引发冲突，故在程序比较复杂、包含更多函数的图形界面程序中推荐使用 UserData 属性。UserData 属性，用户数据属性中的属性值，可以在函数之间或者递归函数内部的不同部分之间传递。前面章节已经介绍 UserData 属性的取值为一个矩阵，通过它来保存与图形对象有关的数据和信息，MATLAB 本身并不使用这些数据，但是用户可以通过 Set()函数和 Get()函数来设置和获得它们。

下面例子 f9_7.m 为使用 UserData 属性改写例子 f9_6.m。

f9_7.m

```
function UserDataGUI()
t=0:0.1:2*pi;
s=sin(t);
c=cos(t);
fh=figure('defaultAxesColor',[.6 .6 .6]); %设置坐标轴的初始颜色值
%--------------------------------------------------------------
AH1=subplot(2,2,1);
grid on
set(fh,'UserData',[AH1]); %把句柄值 fh 存放在图形窗口对象的 UserData 属性中
set(get(fh,'UserData'),'DefaultLineLineStyle','-');
                                    %设置第一个坐标轴的默认线型（第二坐标轴同）
set(get(fh,'UserData'),'LineWidth',1);
                                    %设置第一个坐标轴的线的宽度（第二坐标轴同）
set(set(fh,'UserData'),'Color',[.9 .9 .9]);
                                    %设置第一个坐标轴的前景颜色（第二坐标轴同）
line('Xdata',t,'Ydata',s)
%----------------------------------------------------------------------
AH2=subplot(2,2,2);
grid on;
set(fh,'UserData',[AH2]);
set(get(fh,'UserData'),'DefaultLineLineStyle','-.');
set(get(fh,'UserData'),'LineWidth',2);
set(set(fh,'UserData'),'Color',[.8 .8 .8]);
line('Xdata',t,'Ydata',c)
%-------------------------------------------
```

函数执行结果如图 9-42 所示。

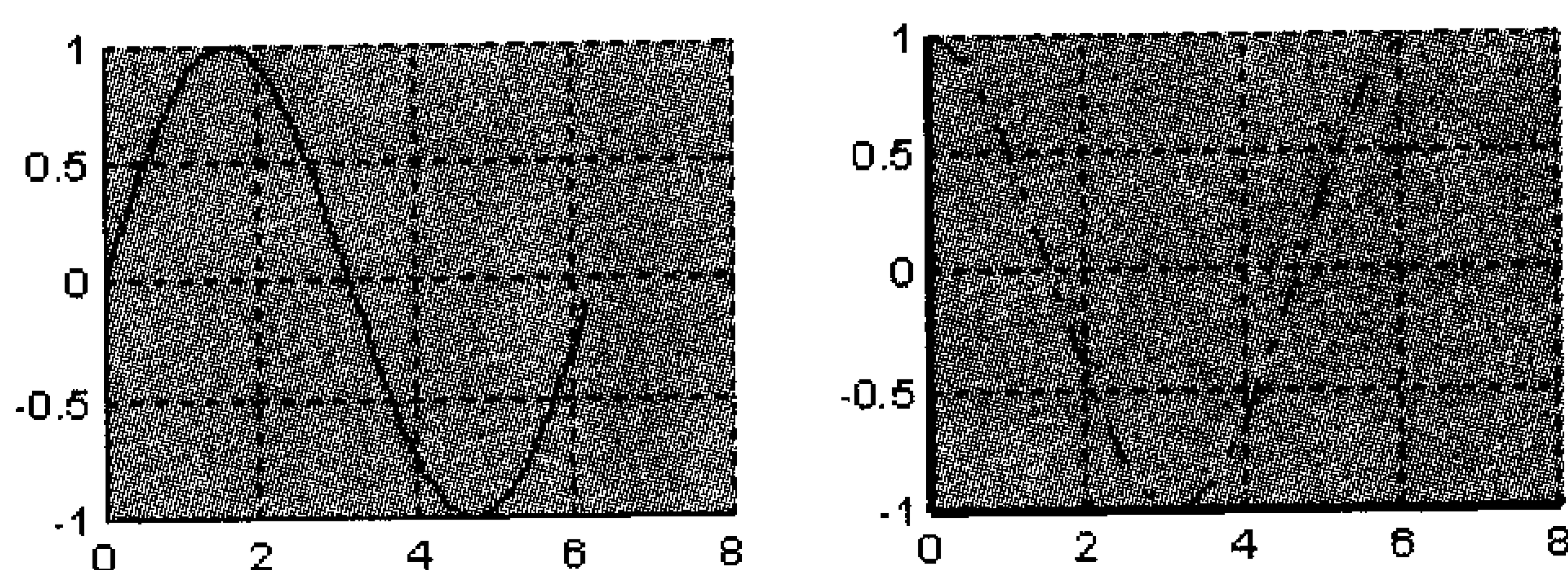


图 9-42　采用 UserData 的函数显示的图形

9.6.3　脚本式 M-file 编程

脚本式 M-file 包含了各种命令、变量，它们都在 MATLAB 的命令窗口中执行。脚本式函数的使用大大减轻了在命令行逐条执行的烦琐，提高了编程效率。

例子 f9_8.m 采用脚本式 M 文件的方式创建一个图形对象，通过与弹出菜单的交互来显示不同的函数图形。

f9_8.m

```
h0=figure('toolbar','none',...
    'position',[200 150 450 250],...
    'name','The Pop-menu');            %创建一个图形窗口对象，并设置位置和标题属性
x=0:0.5:2*pi;
y=sin(x);
h=plot(x,y);
grid on
hm=uicontrol(gcf,'style','popupmenu',...    %创建一个弹出菜单对象
    'string',...
    'sin(x)|cos(x)|sin(x)+cos(x)|exp(-sin(x))',...
    'position',[250 20 50 20]);
set(hm,'value',1)
cab1=[...                                    %编写菜单的回调程序
        'v=get(hm,''value'');,',...
         'switch v,',...
        'case 1,',...
            'delete(h),',...
            'y=sin(x);,',...
            'h=plot(x,y);,',...
            'grid on,',...
       'case 2,',...
            'delete(h),',...
            'y=cos(x);,',...
            'h=plot(x,y);,',...
            'grid on,',...
       'case 3,',...
            'delete(h),',...
            'y=sin(x)+cos(x);,',...
```

```
                'h=plot(x,y);,',...
                'grid on,',...
            'case 4,',...
                'delete(h),',...
                'y=exp(-sin(x));,',...
                'h=plot(x,y);,',...
                'grid on,',...
            'end'];
set(hm,'callback', cab1)
set(gca,'position',[0.2 0.2 0.6 0.6])
title(' The Pop-menu ');
```

运行程序结果显示如图 9-43 所示。

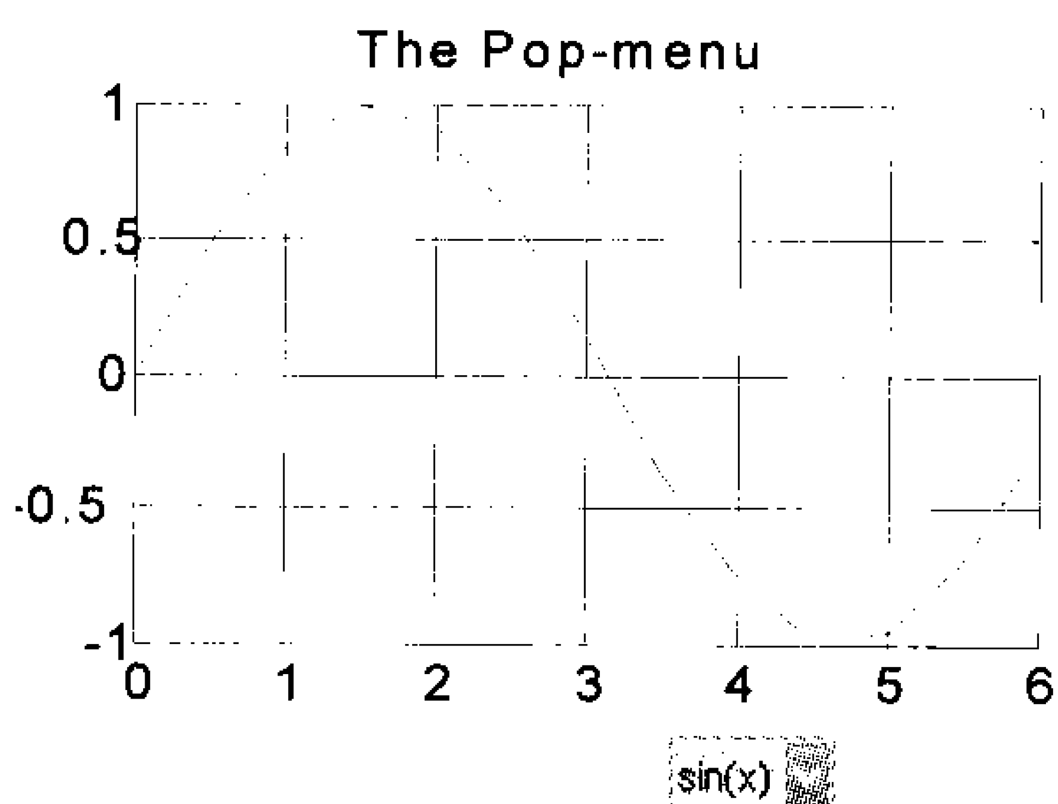


图 9-43　命令行方式创建弹出式菜单

9.6.4　函数式 M-file 编程

函数式 M-file 相对于脚本式 M-file 来说，更加灵活、方便，功能更加强大。它可以有输入参数和返回值，其他函数可以自由调用它。函数式 M-file 中的所有变量如果没有特别声明为全局变量时，都是临时变量。所有的临时变量只在函数本身内有效，这样就可以有效避免变量之间的冲突。下面举例说明函数式 M-file 的使用方法。

我们再来看看 9.1 节中所提到的例子 slider_gui ( )，这是一个典型的函数式 M-file 编程的 GUI 程序。

本例中运用函数分别创建了一个图形界面、一个滑动条和文本输入框以及静态文本标签。通过编写回调函数实现以下两个主要功能。

- 在文本框中输入数值改变滑动条的位置。
- 用鼠标单击或者拖动滑动条来改变其位置，而其位置参数显示在文本框中。

函数代码如下。

f9_9.m

```
function slider_gui                          %创建主函数，函数式 M-file 第一行总以“function”
开始
fh = figure('Position',[250 250 350 350]);   %创建图形界面窗口，并定义其位置和标题名
```

```
    %--------采用 Uicontrol()函数在新建的图形窗口上创建三个控件对象滑动条、文本框和静态文本标签---------------
    sh = uicontrol(fh,'Style','slider',...      %创建滑动条控件，并指定滑动条的参数值，最大为 100，最小为 0
                'Max',100,'Min',0,'Value',25,...     %默认值为 25
                'SliderStep',[0.05 0.2],...          %设置滑动的步长
                'Position',[300 25 20 300],...       %滑动条在图形窗口中的位置
                'Callback',@slider_callback);        %设置 callback 属性为调用子函数 slider_callback

    eth = uicontrol(fh,'Style','edit',...        %创建文本输入框，并设置相关属性，callback 属性为调用
                                                 %子函数 edittext_callback
                'String',num2str(get(sh,'Value')),...
                'Position',[30 175 240 20],...
                'Callback',@edittext_callback);

    sth = uicontrol(fh,'Style','text',...                  %创建静态文本标签，并设置相关属性
                'String','Enter a value or click the slider.',...
                'Position',[30 215 240 20]);
    number_errors = 0;
    previous_val = 0;
    val = 0;

    % -------编写 slider 的触发回调程序，把滑动条的属性值赋值给文本编辑框的 String 属性--------
      function slider_callback(hObject,eventdata)
        previous_val = val;
        val = get(hObject,'Value');  %获取当前对象的'Value 值，并赋给 Val'
        set(eth,'String',num2str(val)); %滑动的距离数值由数值型转变成字符串型然后存入文本框的'String'属性中
        sprintf('You moved the slider %d units.',abs(val - previous_val)) %在 Matlab 主窗口显示移动的单元数
      end
    % -----------------------------------------------------------------------------------------------

    %-------------------------编写 edittext 文本输入框的的触发回调程序------------------------
      function edittext_callback(hObject,eventdata)
          previous_val = val;
          val = str2double(get(hObject,'String'));      %把从文本框获得的数据转变为双精度型赋值给 Val
              %然后判断 val 是否符合输入数值范围要求，符合则赋值给滑动条的‘Value’属性值，用以来改变滑动条的位置，然后在 Matlab 主窗口显示滑动条移动的单元数
          if isnumeric(val) && length(val) == 1 && ...
              val >= get(sh,'Min') && ...
              val <= get(sh,'Max')
              set(sh,'Value',val);
              sprintf('You moved the slider %d units.',abs(val - previous_val))
          else
          %如果输入的数据不符合输入要求则统计出错次数，并在文本框中显示出错次数信息
              number_errors = number_errors+1;
              set(hObject,'String',...
```

```
                    ['You have entered an invalid entry ',...
                    num2str(number_errors),' times.']);
                val = previous_val;
            end
        end
    end
```

运行程序结果显示如图 9-44 所示。

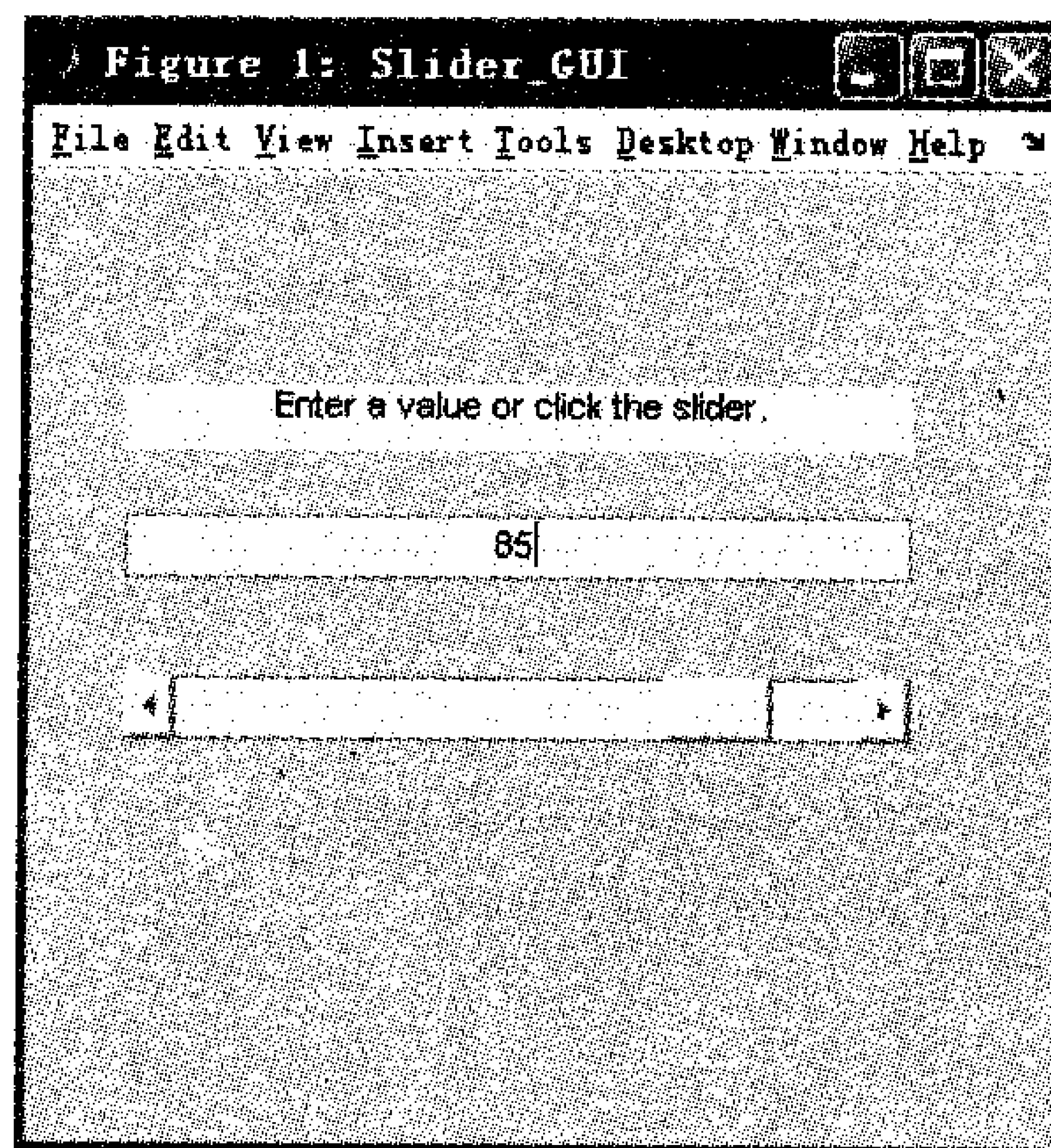


图 9-44　函数式 M-file 程序编写的 GUI

9.7　图形用户界面设计原则和一般步骤

9.7.1　GUI 的设计原则

在 MATLAB R2007 中，GUI 的设计原则和一般的动态界面设计原则基本上都是一致的，它们都强调以下几点。

（1）关注用户及其任务，而不是技术。

设计图形用户界面的目的就是为了帮助用户简化操作，让用户尽可能少花费时间而更加流畅地执行多的任务。所以，GUI 设计者要更多地关注用户的需求，把任务设计得更为用户所理解，不要单纯地追求某种技术而忽视用户需求。

（2）功能优先，然后才是表示。

坚持实现界面程序的功能放在第一位，不能为了表示方便或者界面更加美观而牺牲或削弱部分功能。

（3）从用户的视角看问题，使用用户的词汇进行描述。

现在任何产品都在讲究为客户服务，GUI 设计也不例外，必须从客户的角度看待问题，而不是程序开发者，程序中涉及的用语和词汇都要从习惯用语和词汇出发来考虑。

（4）不要向用户暴露实现细节。

图形用户界面的特点就是避免用户和程序的动态交互中烦琐的细节，所以，GUI 程序

不必公布程序具体的执行细节，只需告诉用户做什么就可以。

（5）用户任务简单化，不要让用户解决额外的问题。

这一原则主要是告诫 GUI 设计者不能把任务设计得太庞大、复杂，太复杂的任务不利于用户快速了解任务的功能和目的。不要让用户为该程序而过多地花时间去解决与程序相关的额外的问题。

（6）保持一致性，引导用户的使用习惯。

图形用户界面程序的一致性一般主要指的是图形界面的菜单设置、控件的位置和大小设置、字体的大小和颜色、窗口的大小，以及按钮的设置等风格一致。

（7）保持显示惯性，传递信息，而不仅仅是数据。

（8）设计应满足响应需求。

整个 GUI 设计要满足程序的需求，程序不能忽视或设计成某个菜单或控件没有响应，而且响应必须是和预期一致的，出现的响应必须是一致的结果，确保程序的健壮性。

9.7.2　GUI 设计的一般步骤

图形用户界面设计的步骤一般包括以下几点。

（1）设计用户界面大致风格。

（2）添加用户界面程序需要的组件。

（3）设置各组件的属性。

（4）编写回调函数。

（5）调试。

下面从具体的案例中来看 GUI 设计的一般步骤。

在实例 Sample_GUI 中，设计了一个图形显示的 GUI 程序。创建一个弹出菜单，把函数 Peaks、membrane 和 sinc 分别作为菜单的菜单项，用户选择不同的显示按钮时，3 个函数值分别以 Surf、Mesh 和 Contour 不同的图形方式显示出来。

第一步：在命令窗口输入命令 GUIDE 或者执行【File】→【New】→【GUI】命令启动对象编辑器（如图 9-45 所示），然后对自己要设计的图形用户界面进行总体布局。

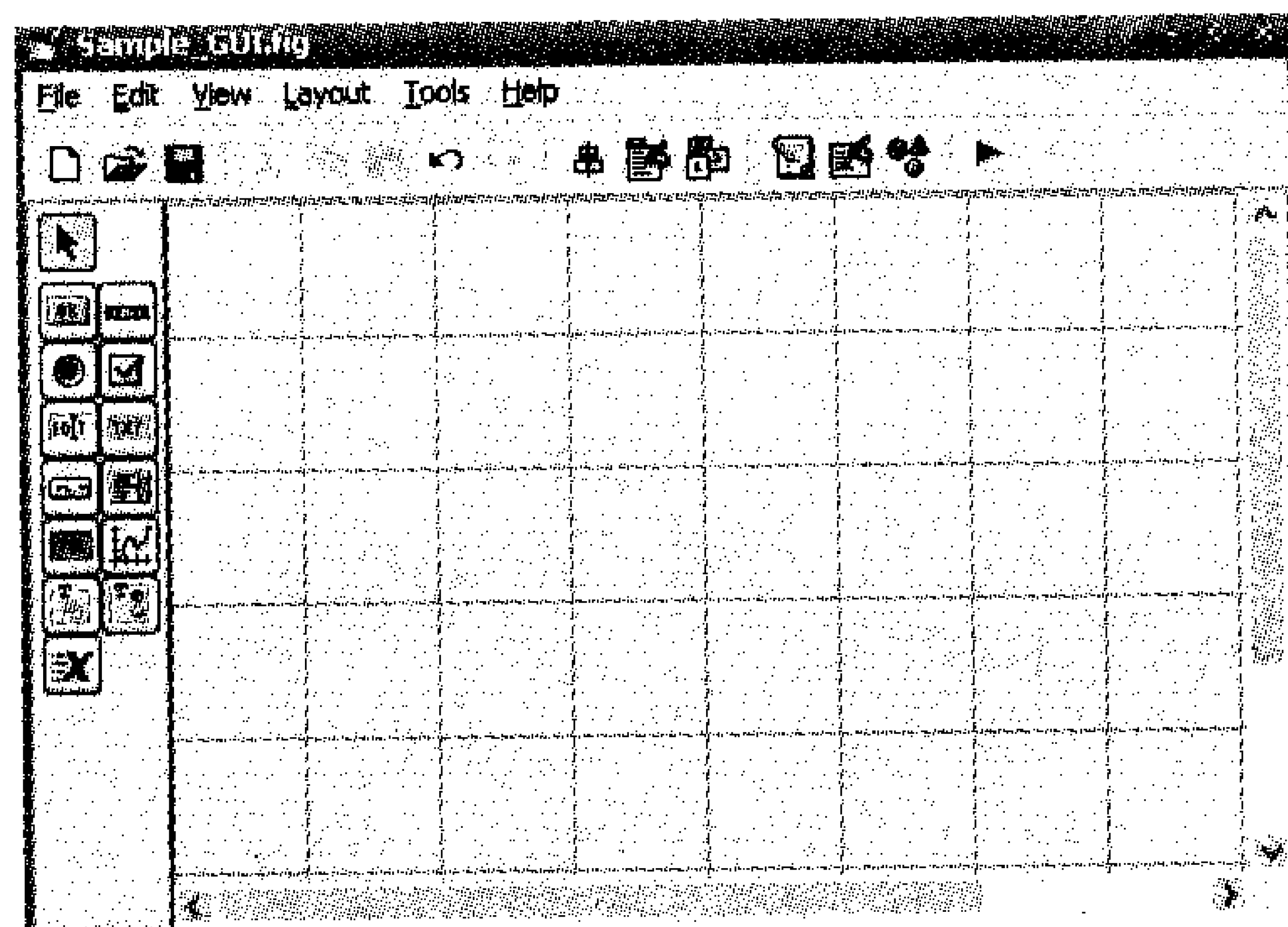


图 9-45　启动对象编辑器

第二步：添加需要的控件对象、菜单对象或者坐标轴对象。在这个例子中，我们需要

添加 1 个坐标轴对象、3 个命令按钮、1 个静态文本标签和 1 个弹出菜单。用鼠标单击对象编辑器左边的控件栏添加相关的组件对象，如图 9-46 所示。

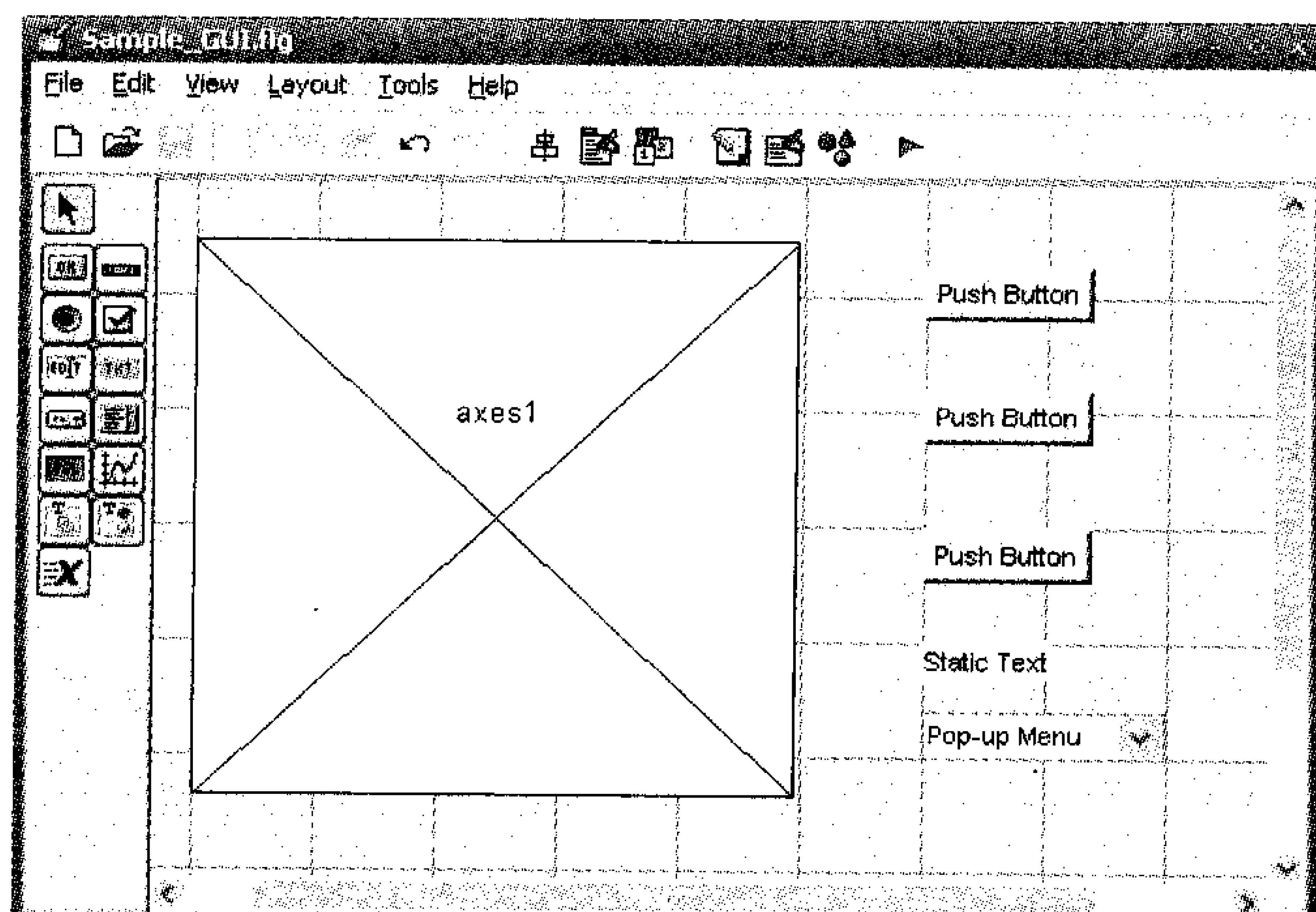


图 9-46　为图形用户界面添加了组件对象

第三步：修改对象属性、调整对象的位置。选择对象属性编辑器来查看和修改控件对象的属性，然后通过对象位置调整工具调整对象位置使得界面更加规范、美观，如图 9-47 所示。

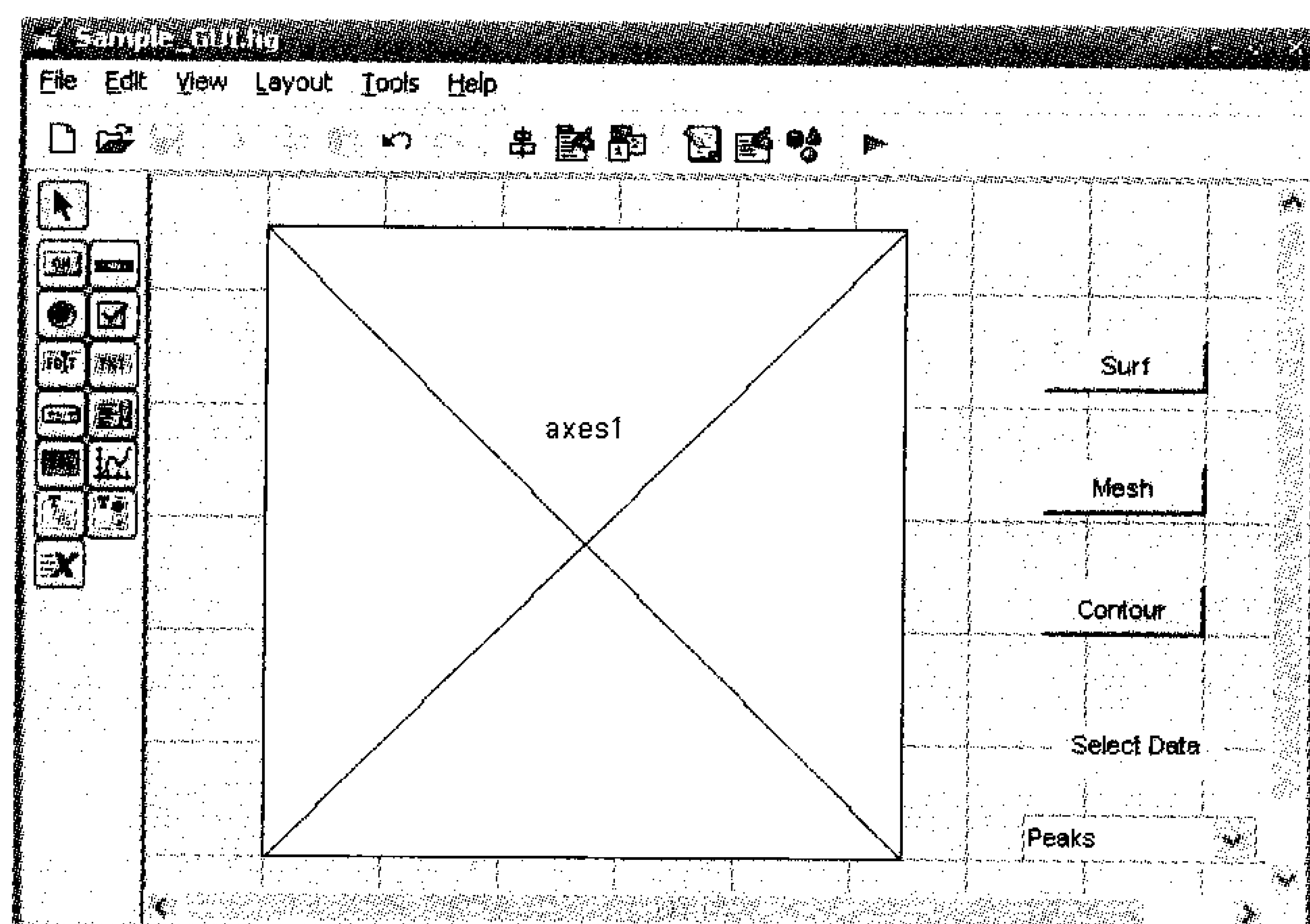


图 9-47　修改对象属性后的图形界面

第四步：设置好界面的各个对象以后，图形用户界面程序设计最重要、也是最关键的一步就是编写好回调程序，实现图形用户界面程序的功能。

首先，在公共函数中添加如下代码：

```
handles.peaks=peaks(35);   %使用函数 peak、membrane、sinc 产生数据并存储在图形对象的句柄
                           %中，作为参数传送给回调程序
handles.membrane=membrane;
[x,y] = meshgrid(-8:.5:8);
r = sqrt(x.^2+y.^2) + eps;
sinc = sin(r)./r;
```

```
handles.sinc = sinc;
handles.current_data = handles.peaks;
surf(handles.current_data);
```

其次，再添加各个命令按钮的 Callback 程序代码，分别在 Surf、Mesh 和 Contour 3 个命令按钮的 callback 中添加如下代码：

```
surf(handles.current_data);          %以 surf 图形显示
mesh(handles.current_data);          %以 mesh 图形显示
contour(handles.current_data);       %以 contour 图形显示
```

最后，编写弹出菜单 Popumenu 的回调程序：

```
str = get(hObject, 'String');
val = get(hObject,'Value');
switch str{val};
case 'Peaks'                                %选择 peaks（）函数.
        handles.current_data = handles.peaks;
case 'Membrane'                             %选择 membrane（）函数
        handles.current_data = handles.membrane;
case 'Sinc'                                 %选择 sinc（）函数
        handles.current_data = handles.sinc;
end
guidata(hObject,handles);
```

程序的详细代码请参见 f9-10.m

第五步：调试。编写好所有回调程序以后，单击 M-file 编辑器中的菜单【Debug】→【Run】调试程序。运行结果如图 9-48 所示。

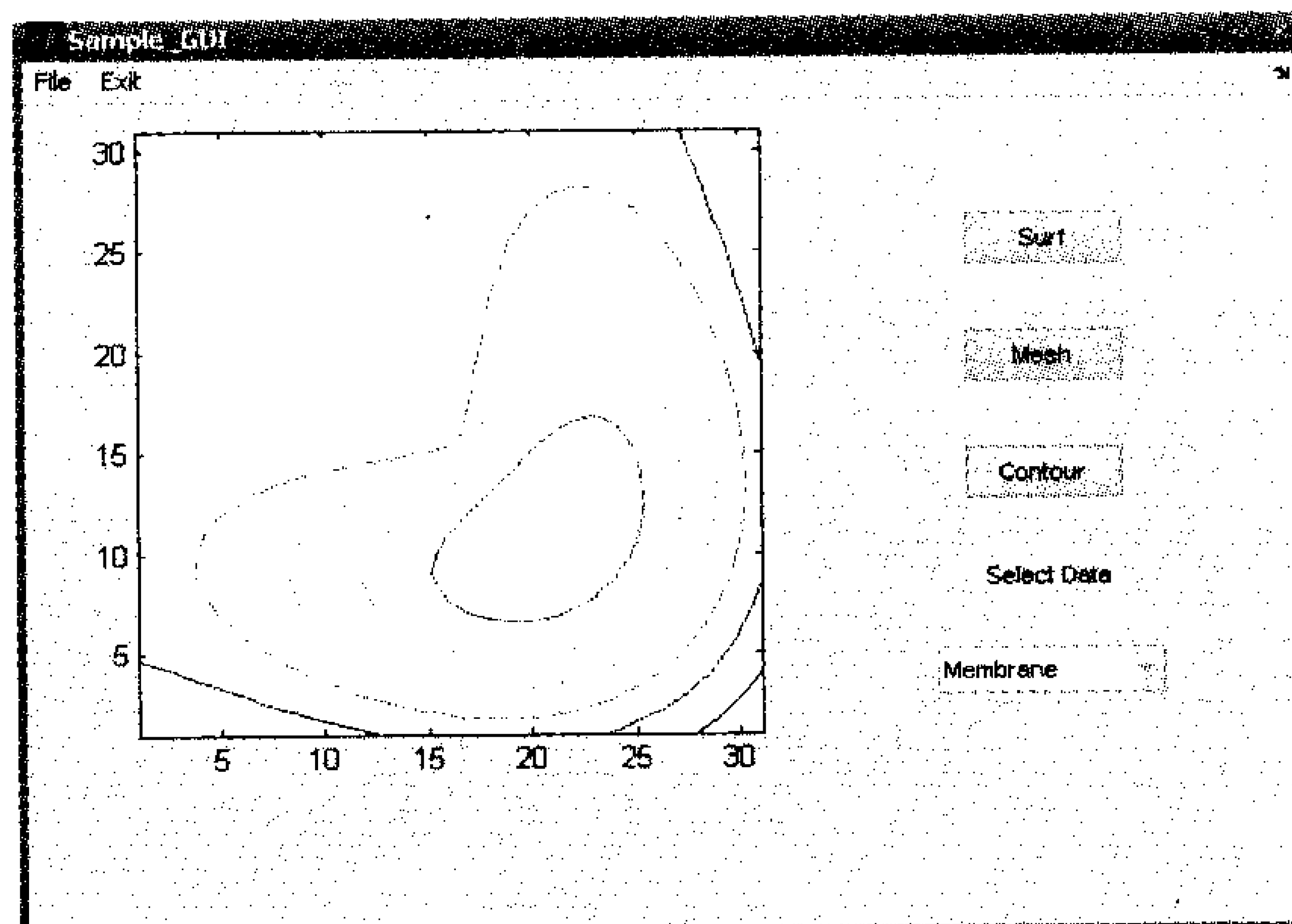


图 9-48　实例运行效果图

9.8　图形用户界面设计实例

至此，有关图形用户界面程序设计的环境和各个对象的用途、属性以及图形编程都做了相关的介绍。这一节将介绍一个综合实例，希望对读者在图形用户界面程序设计方面有所帮助。

该图形界面程序主要实现图像的边缘检测功能。程序采用不同的边缘检测方法来处理

各种图像，用户通过弹出菜单来选择不同的边缘检测方法和不同的图像，并且通过设置不同的属性值来控制不同显示方法的参数。

1．界面设计

本 GUI 程序采用函数式 M-file 创建图形界面，在界面上创建两个用于显示原始图像和边缘检测以后图像的 Axes 对象；然后创建 3 个框架，在第一个框架内创建两个弹出菜单，分别用于选择不同的图片和边缘检测方法，在第二个框架内创建两个单选按钮用于指定阈值的大小，创建一个弹出菜单用于指定边缘检测的方向，第三个框架内创建 3 个按钮，功能分别为边缘计算、显示相关信息以及关闭 GUI 程序。界面设计效果如图 9-49 所示。

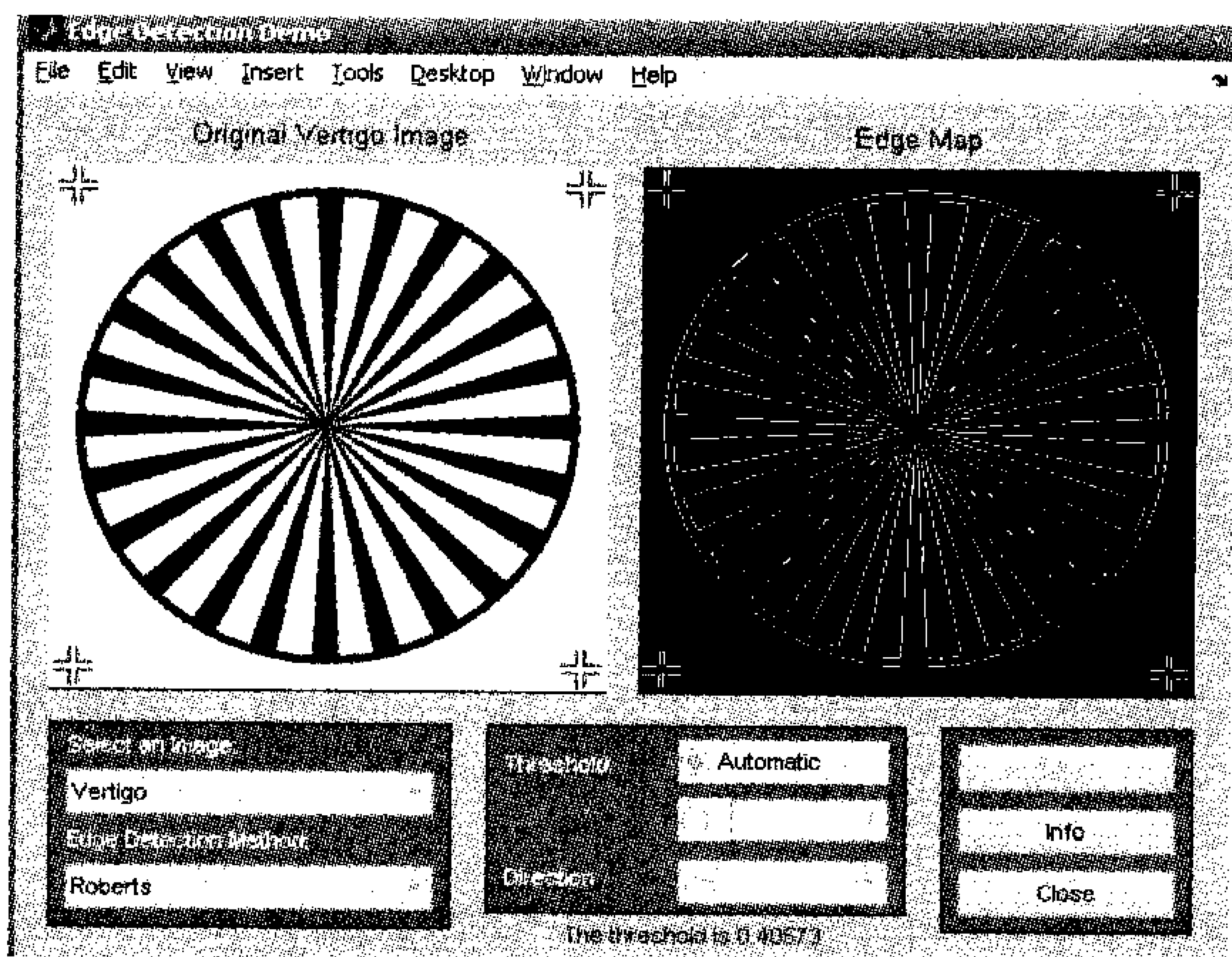


图 9-49　边缘检测程序运行界面

2．编写实现各个功能的回调函数

（1）首先，创建主函数 edgedemo()。

```
function edgedemo(action, varargin)
if nargin<1,
    action='InitializeEDGEDEMO';
end;
feval(action,varargin{:});
return;
```

（2）编写该 GUI 程序主要的子函数 InitializeEDGEDEMO，初始化图形窗口的控件、坐标轴和图形对象。

```
%-------------------------子函数 - InitializeEDGEDEMO ----------------------------
function InitializeEDGEDEMO()
h = findobj(allchild(0), 'tag', 'Edge Detection Demo');
if ~isempty(h)
    figure(h(1))
    return
end
screenD = get(0, 'ScreenDepth');
if screenD>8
    grayres=256;
```

```
    else
        grayres=128;
    end
%-------------创建图形窗口对象并设置相关属性----------------------
    EdgeDemoFig = figure( ...
        'Name','Edge Detection Demo', ...
        'NumberTitle','off', 'HandleVisibility', 'on', ...
        'tag', 'Edge Detection Demo', ...
        'Visible','off', 'Resize', 'off',...
        'BusyAction','Queue','Interruptible','off', ...
        'Color', [.8 .8 .8], ...
        'IntegerHandle', 'off', ...
        'DoubleBuffer', 'on', ...
        'Colormap', gray(grayres));
```

%---获得图形窗口的位置,并判断屏幕尺寸是否足够大来显示图形窗口，窗口太大则删除图形，否则调整窗口大小---

```
    figpos = get(EdgeDemoFig, 'position');
    figpos(3:4) = [560 420];
    horizDecorations = 10;
    vertDecorations = 45;
    screenSize = get(0,'ScreenSize');
    if (screenSize(3) <= 1)
        screenSize(3:4) = [100000 100000];
    end
    if (((figpos(3) + horizDecorations) > screenSize(3)) | ...
                ((figpos(4) + vertDecorations) > screenSize(4)))
        delete(EdgeDemoFig);
        error(['Screen resolution is too low ', ...
                '(or text fonts are too big) to run this demo']);
    end
    dx = screenSize(3) - figpos(1) - figpos(3) - horizDecorations;
    dy = screenSize(4) - figpos(2) - figpos(4) - vertDecorations;
    if (dx < 0)
        figpos(1) = max(5,figpos(1) + dx);
    end
    if (dy < 0)
        figpos(2) = max(5,figpos(2) + dy);
    end
    set(EdgeDemoFig, 'position', figpos);

    rows = figpos(4); cols = figpos(3);
    hs = (cols-512) / 3;            % 水平距离
    bot = rows-2*hs-256;            %图像的底部宽度
    %--------------------------------------------------
    ifs = hs/2;
    Std.Interruptible = 'off';
    Std.BusyAction = 'queue';
```

%---创建放置原始图像的坐标轴对象，并设置相关属性，利用 Set()函数设置显示图像的标题---

```
    hdl.ImageAxes = axes(Std, ...
```

```
            'Units', 'Pixels', ...
            'Parent',EdgeDemoFig,...
            'ydir', 'reverse', ...
            'XLim', [.5 256.5], ...
            'YLim', [.5 256.5],...
            'CLim', [0 255], ...
            'Position',[hs bot 256 256], ...
            'XTick',[],'YTick',[]);
        set(get(hdl.ImageAxes, 'title'), 'string', 'Original Image');
%---创建图形边缘检测的坐标轴，同样利用 Set()函数设置显示图像的标题----------
        hdl.EdgeAxes = axes(Std, ...
            'Units', 'Pixels', ...
            'Parent',EdgeDemoFig,...
            'ydir', 'reverse', ...
            'XLim', [.5 256.5], ...
            'YLim', [.5 256.5],...
            'CLim', [0 1], ...
            'Position',[cols-hs-256 bot 256 256], ...
            'XTick',[],'YTick',[]);
        set(get(hdl.EdgeAxes, 'title'), 'string', 'Edge Map');
%-------------原始图像-----------
        hdl.Image = image(Std, ...
            'CData', [], ...
            'CDataMapping', 'scaled', ...
            'Parent',hdl.ImageAxes,...
            'Xdata', [1 256],...
            'Ydata', [1 256],...
            'EraseMode', 'none');
%---------------- 边缘图像------------------
        hdl.Edge = image(Std, ...
            'CData', [], ...
            'CDataMapping', 'scaled', ...
            'Parent',hdl.EdgeAxes,...
            'Xdata', [1 256],...
            'Ydata', [1 256],...
            'EraseMode', 'none');

        bgcolor = [0.45 0.45 0.45];   %设置框架的背景颜色
        fgcolor = [1 1 1];                %设置文本颜色
%----------创建放置选择的图像和检测方法的菜单的框架------------
        mfleft=hs;
        mfbot=hs;
        mfwid=(3*cols/8)-1.5*hs; % 2*cols/7
        mfht=bot-2*hs;
        hdl.MenuFrame = uicontrol(Std, ...
            'Parent', EdgeDemoFig, ...
            'Style', 'frame', ...
            'Units', 'pixels', ...
            'Position', [mfleft mfbot mfwid mfht], ...
            'BackgroundColor', bgcolor);
```

%---创建选择不同图像的弹出菜单，并设置 Callback 属性为调用 LoadNewImage 子函数---

```
ipwid = mfwid-2*ifs;
ipht = 21;
ipleft = mfleft+ifs;
ipbot = mfbot+1.7*ifs + 2*ipht;
hdl.ImgPop=uicontrol(Std, ...
    'Parent', EdgeDemoFig, ...
    'Style','popupmenu', ...
    'Units','pixels', ...
    'Position',[ipleft ipbot ipwid ipht], ...
    'Enable','on', ...
    'String','Coins|Circuit|Vertigo|Lifting Body|Rice|Saturn|Eight Bit|Glass', ...
    'Tag','ImagesPop',...
    'Callback','edgedemo(''LoadNewImage'')');

uicontrol( Std, ...                    %添加弹出菜单中图像的解释文本标签
    'Parent', EdgeDemoFig, ...
    'Style','text', ...
    'Units','pixels', ...
    'Position',[ipleft ipbot+ipht ipwid 18], ...
    'Horiz','left', ...
    'Background',bgcolor, ...
    'Foreground',fgcolor, ...
    'String','Select an Image:');
```

% 创建选择不同边缘检测方法的弹出菜单，并设置 Callback 属性为调用 SelectMethod 子函数

```
hdl.Method = 'Sobel';
mpwid = ipwid;
mpht = ipht;
mpleft = ipleft;
mpbot = mfbot+1.2*ifs;
hdl.MethodPop=uicontrol(Std, ...
    'Parent', EdgeDemoFig, ...
    'Style','popupmenu', ...
    'Units','pixels', ...
    'Position',[mpleft mpbot mpwid mpht], ...
    'Enable','on', ...
    'String','Sobel|Prewitt|Roberts|Laplacian of Gaussian|Canny', ...
    'Tag','MethodPop',...
    'Callback','edgedemo(''SelectMethod'')');

uicontrol( Std, ...
    'Parent', EdgeDemoFig, ...
    'Style','text', ...
    'Units','pixels', ...
    'Position',[mpleft mpbot+mpht mpwid 18], ...
    'Horiz','left', ...
    'Background',bgcolor, ...
```

```
        'Foreground',fgcolor, ...
        'String','Edge Detection Method:');    %添加边缘检测方法选择解释的文本标签
%------------------创建参数设置的框架，在这里可以指定边缘检测方法的参数
--------------------------
    pfleft =(3*cols/8)+0.5*hs;
    pfbot = 1.5*hs;
    pfwid =(3*cols/8)-hs;
    pfht = bot-2.5*hs;
    hdl.ParamFrame = uicontrol(Std, ...
        'Parent', EdgeDemoFig, ...
        'Style', 'frame', ...
        'Units', 'pixels', ...
        'Position', [ pfleft pfbot pfwid pfht ], ...
        'BackgroundColor', bgcolor);
    %------------------------------------------------------------------------------
    labelleft = pfleft+ifs;
    labelwid = pfwid/2-hs;
    labelbot = pfbot+2*pfht/3;
    hdl.sprThLbl = uicontrol(Std,...      %建立阈值控制的文本标签
        'Parent', EdgeDemoFig, ...
        'Style','text', ...
        'Units','pixels', ...
        'Position',[labelleft labelbot labelwid 18], ...
        'Horiz','left', ...
        'String','Threshold:', ...
        'BackgroundColor',bgcolor, ...
        'ForegroundColor',fgcolor);
    hdl.Threshold = 0;       %初始化阈值

    raleft = pfleft + pfwid/2 - hs/2;
    rabot = pfbot+2*pfht/3+hs/6;
    rawid = pfwid/2;
    raht = ipht;
    hdl.RadioAutomatic=uicontrol(Std, ...
        'Parent', EdgeDemoFig, ...
        'Style','radiobutton', ...
        'Units','pixels', ...
        'Position',[raleft rabot rawid raht], ...
        'String','Automatic', ...
        'value',1,'Userdata',1, ...
        'Callback','edgedemo(''Radio'',''auto'')');       %调用子函数 Radio,参数为 Auto

    rmleft = pfleft + pfwid/2 - hs/2;
    rmbot = pfbot+pfht/3+hs/3;
    rmwid = hs*1.5;
    rmht = ipht;
    hdl.RadioManual=uicontrol(Std, ...
        'Parent', EdgeDemoFig, ...
        'Style','radiobutton', ...
        'Units','pixels', ...
```

```
        'Position',[rmleft rmbot rmwid rmht], ...
        'String','', ...
        'value',0,'Userdata',0, ...
        'Callback','edgedemo(''Radio'',''manual'')');   %调用子函数 Radio,参数为 manual，数值由用户设置

    thleft = rmleft+rmwid;
    thwid = rawid-rmwid;
    thbot = rmbot;
    thht = rmht;
    hdl.ThreshCtrl = uicontrol(Std, ...
        'Parent', EdgeDemoFig, ...
        'Enable', 'off', ...
        'Style','edit', ...
        'Units','pixels', ...
        'Position',[thleft thbot thwid thht], ...
        'Horiz','right', ...
        'Background','white', ...
        'Foreground','black', ...
        'String','0',...
        'callback','edgedemo(''UpdateSprThresh'')');    % 调用子函数 UpdateSprThresh'
```

%--------- 创建边缘检测方向的弹出菜单，用户可以选择垂直方向或者水平方向检测，callback 属性设置为调用子函数 UpdateDirectionality---------------------

```
    dpwid = pfwid/2;
    dpht = ipht;
    dpleft = pfleft + pfwid/2 - hs/2;
    dpbot = pfbot+.4*hs;
    hdl.sprDirPop=uicontrol(Std, ...
        'Parent', EdgeDemoFig, ...
        'Style','popupmenu', ...
        'Units','pixels', ...
        'Position',[dpleft dpbot dpwid dpht], ...
        'Enable','on', ...
        'String','Both|Horizontal|Vertical', ...
        'Tag','DirectionPop',...
        'Callback','edgedemo(''UpdateDirectionality'')');

    labelleft = pfleft+ifs;
    labelwid = pfwid/2-hs;
    labelbot = dpbot;
    hdl.sprDirLbl = uicontrol( Std, ...       %创建解释菜单的文本标签
        'Parent', EdgeDemoFig, ...
        'Style','text', ...
        'Units','pixels', ...
        'Position',[labelleft labelbot labelwid 18], ...
        'Horiz','left', ...
        'Background',bgcolor, ...
        'Foreground',fgcolor, ...
        'String','Direction:');
    hdl.Directionality = 'both';
    hdl.logSigmaCtrl=uicontrol(Std, ...
```

```
            'Parent', EdgeDemoFig, ...
            'Style','edit', ...
            'Units','pixels', ...
            'Position',[dpleft dpbot dpwid dpht], ...
            'Horiz','right', ...
            'Background','white', ...
            'Foreground','black', ...
            'String','2', ...
            'Tag','DirectionPop',...
            'Visible', 'off', ...
            'Callback','edgedemo(''UpdateLOGSigma'')');
        hdl.logSigmaLbl = uicontrol( Std, ...
            'Parent', EdgeDemoFig, ...
            'Style','text', ...
            'Units','pixels', ...
            'Position',[labelleft labelbot labelwid 18], ...
            'Horiz','left', ...
            'Background',bgcolor, ...
            'Foreground',fgcolor, ...
            'Visible', 'off', ...
            'String','Sigma:');
        hdl.LogSigma = 2;
```

%----创建状态栏，在 MATLAB 中并没有现有的状态条控件，本程序中采用静态文本标签来创建状态栏，并设置状态栏的背景颜色---------------

```
        colr = get(EdgeDemoFig,'Color');
        hdl.Status = uicontrol( Std, ...
            'Parent', EdgeDemoFig, ...
            'Style','text', ...
            'Units','pixels', ...
            'Background', colr, ...
            'Foreground', [.8 0 0], ...
            'Position',[pfleft 2 pfwid 18], ...
            'Horiz','center', ...
            'Tag', 'Status', ...
            'String','Initializing Edge Detection Demo...');
```

%------------- 创建放置 Apply 按钮、Info 按钮以及 Close 按钮的框架---------------

```
        bfleft = (3*cols/4)+.5*hs;
        bfbot = hs;
        bfwid = (cols/4)-1.5*hs;
        bfht = bot-2*hs;
        hdl.ButtonFrame = uicontrol(Std, ... %创建框架，并指定框架的位置、大小的单位以及背景颜色
            'Parent', EdgeDemoFig, ...
            'Style', 'frame', ...
            'Units', 'pixels', ...
            'Position', [ bfleft bfbot bfwid bfht ], ...
            'BackgroundColor', bgcolor);
```

%------------ 创建“Apply”按钮控件，设置按钮相关属性，指定其风格以及设置 callback 属性为调用子函数 ComputeEdgeMap，用来开始计算图像的边缘值-------------------

```
        btnwid = bfwid - 2*ifs;
```

```
btnht =   (bfht-4*ifs)/3;
btnleft = bfleft + ifs;
btnbot = bfbot + bfht - ifs - btnht;
hdl.Apply=uicontrol(Std, ...
    'Parent', EdgeDemoFig, ...
    'Style','pushbutton', ...
    'Units','pixels', ...
    'Position',[btnleft btnbot btnwid btnht], ...
    'Enable','off', ...
    'String','Apply', ...
    'Callback','edgedemo(''ComputeEdgeMap'')');
```

%-----------创建“Info”按钮控件，单击该按钮显示该 GUI 程序的相关信息。设置按钮相关属性，指定其风格以及设置 callback 属性为调用 edgedemo 的帮助文件-----------

```
btnbot = bfbot + bfht - 2*ifs - 2*btnht;
hdl.Help=uicontrol(Std, ...
    'Parent', EdgeDemoFig, ...
    'Style','pushbutton', ...
    'Units','pixels', ...
    'Position',[btnleft btnbot btnwid btnht], ...
    'Enable','off', ...
    'String','Info', ...
    'Callback','helpwin edgedemo');
```

%--------创建“Close”按钮控件，设置按钮相关属性，指定其风格以及 “Close”按钮的 callback 属性为 close(gcbf) ，实现关闭当前图形窗口功能---------------

```
btnbot = bfbot + ifs;
hdl.Close=uicontrol(Std, ...
    'Parent', EdgeDemoFig, ...
    'Style','pushbutton', ...
    'Units','pixels', ...
    'Position',[btnleft btnbot btnwid btnht], ...
    'Enable','off', ...
    'String','Close', ...
    'Callback','close(gcbf)');

set(EdgeDemoFig, 'Userdata', hdl, 'Visible', 'on');
drawnow
LoadNewImage(EdgeDemoFig);        %打开新图像
drawnow
set(EdgeDemoFig, 'HandleVisibility', 'Callback');
set([hdl.Apply hdl.Help hdl.Close] , 'Enable', 'on');   %设置“Apply”、“Info”、“Colse”3个按钮为有效状态
return
```

（3）计算图像的边缘值，该子函数调用函数 EDGE.m 来计算图像的边缘值。在实现边缘检测计算时采用分支结构：当选择 Sobel','Roberts','Prewitt’三者之一时，用户可以用选择水平检测、垂直检测或者二者同时检测；当选择‘Laplacian of Gaussian’或者‘Canny’检测方法的时候，用户可以指定参数 Sigma 的值，其控制着 Gaussian 函数的散开程度。同时过滤器的大小由 EDGE 函数基于 Sigma 的值自动设置。

```
%------------------------子函数- ComputeEdgeMap----------------------------
function ComputeEdgeMap(DemoFig)
```

```
if nargin<1
    callb = 1;
    DemoFig = gcbf;
else
    callb = 0;
end

set(DemoFig,'Pointer','watch');                    %设置图形窗口的鼠标指针为“等待”
setstatus(DemoFig, 'Computing the edge map...');
hdl=get(DemoFig,'Userdata');                       %获取图形用户数据
img = getimage(hdl.Image);

autothresh = get(hdl.RadioAutomatic, 'Value');   %获得默认的阈值
switch hdl.Method
case {'Sobel','Roberts','Prewitt'}
    if autothresh   %如果自动选择阈值按钮为真，把边缘检测的方法和检测方向赋值给图像的阈
值参数
        [edgemap,thresh] = edge(img, hdl.Method, hdl.Directionality);
        setstatus(['The threshold is ' num2str(thresh) '.']); %在状态栏显示相关的数据信息
    else
        edgemap = edge(img, ...
            hdl.Method, hdl.Threshold, hdl.Directionality);
        setstatus(DemoFig, '');
    end

case 'Laplacian of Gaussian'
    if autothresh
        [edgemap,thresh] = edge(img, 'log', [], hdl.LogSigma);
        setstatus(DemoFig, ['The threshold is ' num2str(thresh) '.']);
    else
        edgemap = edge(img, 'log', hdl.Threshold, hdl.LogSigma);
        setstatus(DemoFig, '');
    end
case 'Canny'
    if autothresh
        [edgemap,thresh] = edge(img, 'canny', [], hdl.LogSigma);
        setstatus(DemoFig, ['High threshold is ' num2str(thresh(2)) '.']);
    else
        [edgemap,thresh] = edge(img, 'canny', hdl.Threshold, hdl.LogSigma);
        setstatus(DemoFig, '');
    end
otherwise
    error('EDGEDEMO: Invalid edge detection method.');
end             %若是以上三种情况都不是，则弹出“检测方法错误”的错误信息

set(hdl.Edge, 'CData', edgemap);
set(hdl.Apply, 'Enable', 'off');
set(DemoFig,'Pointer','arrow');       设置图形窗口的指针为“箭头”
drawnow;
```

其他的子函数请参见 f9_11.m。

第 10 章　MATLAB 高级接口技术

MATLAB 提供了十分丰富的外部接口，这些接口不仅使得 MATLAB 可以十分方便地与其他应用程序交换数据和信息，还实现了与其他程序函数和算法的交互。所以通过 MATLAB 接口编程，可以充分利用现有资源，能更容易地编写出功能强大、结构简洁的应用程序。

MATLAB 的外部接口使 MATLAB 具备了可以调用 C 和 Fortran 等语言的函数和算法、可以使用 COM 和 DDE 编程技术、可以利用 Internet 上的 Web 服务、可以与计算机硬件交互等功能；同时也使得其他应用程序，如 Excel、Java、.NET 程序可以调用 MATLAB 的函数和算法。

MATLAB 的外部接口使得 MATLAB 可与外部设备和程序实现数据交互和程序移植，可以扩充 MATLAB 强大的数值计算和图形显示功能，从而弥补了其执行效率较低的缺点，同时增强了其他应用程序进行软件开发的功能，改善了软件开发效率。

本章在详细地介绍了 MEX 文件的编写、计算引擎的调用方法和 COM 和 DDE 编程的同时，还介绍了 MATLAB 的硬件接口和 Web 服务，最后以实例介绍了 MATLAB 的 Excel、Java 和.NET 生成器的用法。

本章主要内容：

- MAT 文件和 MEX 文件
- MATLAB 计算引擎
- MATLAB 编译器
- COM 与 DDE 编程
- MATLAB 硬件接口
- MATLAB Web 服务
- Excel、Java、.NET 生成器

10.1　MATLAB 外部接口概述

本节主要介绍 MATLAB 接口的概念，以及 MEX 文件、MAT 文件、MATLAB 计算引擎、MATLAB 编译器、MATLAB COM 和 DDE 编程、MATLAB 硬件接口等概念。通过本节的学习，可对 MATLAB 外部接口有个总体轮廓的认识。

10.1.1　外部接口概述

MATLAB 作为一种“便笺式”程序设计语言，其强大的数值计算和杰出的绘图功能、大量的函数库和高效简洁的编程语言，使其成为非常优秀的数值计算软件，但是 MATLAB

若没有实现外部接口，仍存在以下几个缺点。

- MATLAB 是解释性语言，边解释边执行，使得执行效率不是很理想。实现相同的功能，与 C 语言等相比执行时间相对较长。尤其在进行大规模的数值计算和分析时，MATLAB 有点差强人意，限制了其在这方面的应用。
- MATLAB 程序员将无法调用外部大量已有的 C 或 Fortran 等语言编写的算法，而若用 MATLAB 重新编写，一般效率较低而且费时费力。
- Visual C++的 MFC 类库功能强大，而且编程高效，若能与 Visual C++实现联合编程，将给 MATLAB 程序员带来很大便利。
- MATLAB 的程序文件是 M 文件，脱离 MATLAB 环境无法执行用。而且任何文本编辑文件可以打开编辑 M 文件，不利于代码的保密。
- MATLAB 的数据文件格式为 MAT 文件，并且对于不同的硬件平台和操作系统，文件格式也有不同，使得 MATLAB 与外部环境进行数据交互不方便。
- MATLAB 在数值计算，计算机仿真和绘图方面有着明显的优势，若其他的应用程序中能够调用这些功能，将在很大程度上提高效率。

MATLAB 丰富的外部接口很好地解决了这些问题。通过 MATLAB 的外部接口，在 C 语言和 Java 语言等程序中可以调用 MATLAB 函数，也可以在 MATLAB 程序中调用 C 语言、Fortran 语言等已有的算法。MATLAB 还可以直接访问外围设备，如 modem，打印机以及连接在计算机串口上的其他设备。

具体来说，MATLAB 主要提供了 MEX 文件、MAT 文件、计算引擎、COM 和 DDE 编程、Web 服务、硬件接口和 Excel、Java、.NET 生成器等形式的接口。其中 MEX 文件使 MATLAB 可以调用 C 或 Fortran 语言的函数和算法，同时费时的循环等指令可以编写相应的 MEX 文件以提高效率；MAT 文件实现了 MATLAB 与外部环境的数据交互；MATLAB 计算引擎使其他程序，比如 C 语言和 Fortran 语言等程序可以调用 MATLAB 的函数或算法；MATLAB 支持 COM 和 DDE 技术，使 MATLAB 可以利用现有 COM 组件来开发软件；Web 服务使得 MATLAB 可以利用 Internet 上的服务；硬件接口使 MATLAB 可以方便的访问计算机硬件，实现与外设的交互；Excel、Java 和.NET 生成器都是 MATLAB 编译器的扩展，利用这些生成器将 MATLAB 的函数和算法封装成的组件，在相应的 Excel、Java 和.NET 程序中可以直接使用。这些生成器使得外部程序可以方便地利用 MATLAB 的函数和算法。

10.1.2 MEX 文件

因为 MEX 是 MATLAB Executable 的缩写，所以 MEX 文件就是 MATLAB 可执行程序。在 Windows 环境中，MATLAB 7.1 版本（R14SP3）前，MEX 文件的后缀.dll，说明 MEX 文件是一种动态链接库，向用户提供了 MATLAB 与其他语言的接口。在 MATLAB 7.1 版本中 MEX 文件的后缀做了修改，而后缀是和平台相关的，详细内容见 10.2.1 节。

MEX 文件主要的用途有以下几个方面。

- MATLAB 可通过 MEX 方式在 MATLAB 环境中直接调用存在的 C 语言或 Fortran 语言编写的算法，就像调用 MATLAB 的内嵌函数一样方便，不必重新用 MATLAB 语言编写。

- 对于影响 MATLAB 执行效率的 for 和 while 等循环，可用 C 语言或 Fortran 语言编写相同功能的代码，MATLAB 编译成 MEX 文件后便可直接调用，从而解决了 MATLAB 的效率瓶颈问题。
- 通过 MEX 文件可以直接对硬件进行编程，如进行 A/D 和 D/A 中断等。克服了 MATLAB 对硬件访问不足的缺点。

可见，MEX 文件功能强大，正确使用 MEX 文件不仅可以节省大量的编程时间，而且增强了 MATLAB 的应用程序的功能。

10.1.3 MAT 文件

MAT 文件实现了 MATLAB 的程序和文件与其他编程环境的数据交换。在 Windows 操作系统中，其后缀名为.mat。MAT 文件是 MATLAB 默认的数据存储的文件格式，其数据以字节流的方式顺序写入，以二进制格式存储，这种格式为不同平台之间或 MATLAB 与不同应用程序之间共享数据提供了便利。

在可用文本编辑器中打开一个 MAT 文件，查看文件信息，但一般情况下，不需要了解 MAT 文件的格式，因为 MATLAB 本身提供了一些 API 函数来简化 MAT 文件的读取和存储。

如在第 8.1.5 节“数据导入与导出”中已经讲述的，可用“Save”命令将当前工作区内存区中的数据导出到磁盘上，以二进制格式的 MAT 文件形式存储；用“Load”命令将磁盘 MAT 文件中的数据导入到 MATLAB 的基本工作空间中。

10.1.4 MATLAB 计算引擎

MATLAB 引擎函数库是 MATLAB 提供的一组函数库和程序库，这些函数库可以在其他的非 MATLAB 程序中调用。如用户可在自己编写的 C 语言或其他语言应用程序中调用 MATLAB，完成比较复杂的数学计算，在用户启动 MATLAB 引擎时，相当于启动了另外一个 MATLAB 进程，将其在后台运行。MATLAB 引擎的这种工作模式，类似于客户端/服务器模式，MATLAB 引擎实现了其他程序与 MATLAB 进程的交互，完成了二者之间的数据交换和命令传送的任务。

MATLAB 引擎主要有以下几个功能。

- MATLAB 引擎提供了功能强大的数学函数库，特别是矩阵计算，它是 MATLAB 的强项，而且 MATLAB 对其进行了优化。所以在数学计算方面的应用中，调用 MATLAB 的数学函数，可以方便地完成复杂的数学计算。如求矩阵的特征值和求方程组的根，这些功能如用普通的 C 语言或 Fortran 语言来编写将是非常麻烦的，而且需要程序员有深厚的数学功底（如矩阵的特征值的求法），由于 MATLAB 提供了专门的函数，使用引擎仅仅几行指令就可以完成相同的任务。
- 将 Visual C++语言等的通用编程平台与 MATLAB 结合可构建特殊的系统。此类系统可利用 MATLAB 的计算高效和矩阵处理灵活的优点，同时又结合了其他语

言循环处理快和图形界面编程简单的优点。因此，MATLAB 引擎可以简化应用程序的开发，取得事半功倍的效果。

MATLAB 计算引擎几乎可以利用 MATLAB 的全部功能，但是不能脱离 MATLAB 工作环境，即本地计算机必须安装 MATLAB。

10.1.5 MATLAB 编译器

MATLAB 编译器可以将 MATLAB 的 M 文件编译成 C/C++代码，生成的 C/C++代码再用 C/C++编译器编译链接成可执行程序，所以 MATLAB 编译器是实现 MATLAB 混合编程的基础。MATLAB 的 Excel、Java 和.NET 生成器都是 MATLAB 编译器的扩展，都要通过 MATLAB 编译器来将 MATLAB 的函数或算法转化成相应的源代码。生成的源代码创建成组件，就可以在 Excel、Java 和.NET 程序中使用。

使用 MATLAB 编译器可以将 MATLAB 程序转化成独立运行的应用程序或库，即这些应用程序或库可以在没有安装 MATLAB 的环境下正常运行。所以 Excel、Java 和.NET 生成器创建的组件也可以脱离 MATLAB 环境使用。MATLAB 可以编译 M 文件、MEX 文件和其他 MATLAB 代码。MATLAB 编译器支持 MATLAB 的所有特性，包括对象、私有函数和方法。

从 MATLAB 5.3 版本以来 MATLAB 都自带了 C 语言的编译器。但是 MATLAB 编译器不是对所有的 M 文件都可以转换成 C/C++代码，对此特定版本的编译器有着不同的限制，如不能转换脚本 M 文件，只能函数 M 文件。

10.1.6 MATLAB COM 和 DDE 编程

组件对象模型（Component Object Model，COM）提供了一个可将可重用的软件集成到应用程序中的框架。MATLAB 支持 COM 技术，大大简化了应用程序开发，因为组件是以编译后的代码形式的，源码可以由任意支持 COM 的编程语言编写；使用组件的应用程序升级变得简单，因为升级组件时不必重新编译整个应用程序；另外，组件的位置对于应用程序来说是透明的，所以不必修改应用程序即可重新载入组件。使用 COM 技术，开发者和用户可以选择不同厂商生产的组件集成到一起，由此来创建一个应用程序。

MATLAB 提供了使 MATLAB 与 Windows 应用程序相互访问的函数。这些函数使用动态数据交换（Dynamic Data Exchange，DDE），DDE 允许 Windows 应用程序之间以交换数据形式通信。

10.1.7 MATLAB Web 服务

MATLAB Web 服务包括基于 XML 的一套技术，可以通过网络调用远程程序。网络指当地局域网或万维网。简而言之，Web 服务可使的程序在不同的操作系统或平台之间实现交互。MATLAB 实现了 SOAP（Simple Object Access Protocol）和 WSDL（Web Services Description Language）两种 Web 服务技术，MATLAB 可作为 Web 服务的客户端，向服务

器发送请求并处理响应。

使用 Web 服务，MATLAB 可以利用 Internet 上的资源，如 Internet 上的 MATLAB 函数或算法。但是如同 Internet 一样，Web 服务是不断发展变化的，也是不稳定的，所以有 Web 服务的 MATLAB 应用程序不能保证其可靠性，在开发时要尤为注意。

10.1.8 Excel、Java 和.NET 生成器

Excel、Java 和.NET 生成器都是 MATLAB 编译器的扩展。在 MATLAB R2007a 版本中，MATLAB 将 Excel、Java 和.NET 生成器以及 MATLAB 编译器开发环境集成到了一起，图形界面工具可以使用配置工具（Deploytool），命令行方式可使用 mcc 命令。即创建 Excel、Java 和.NET 生成器组件和独立运行程序或 C/C++库都可以用配置工具或 mcc 命令来完成。若创建后还需打包和发布则必须使用配置工具。

Excel、Java 和.NET 生成器可以将 MATLAB 的函数或算法封装成组件，根据需要创建的组件可以在 Excel 环境、Java 应用程序和 .NET 应用程序中使用。必要时可将创建的组件和运行所需文件打包发布给终端用户，使得用户可在没有安装 MATLAB 的机器上正常使用创建的组件。

10.1.9 MATLAB 硬件接口

MATLAB 提供了访问硬件的接口函数与方法，使得 MATLAB 可以和计算机外设方便地进行通信，大大扩展了 MATLAB 的应用领域。MATLAB 与硬件接口包括串口通信和并口通信两种。本书主要介绍 MATLAB 的串口通信，MATLAB 串口接口提供了与外围设备的直接连接，如调制解调器（Modems）、打印机和连接到机器串口上的其他科学设备。

10.1.10 MATLAB 外部接口

使用 MATLAB 外部接口，首先，要选择接口的使用方式（根据 MATLAB 提供的多种支持方式，选择一种最适合的应用方式）。其次，需要选择应用程序的开发语言。最后，选择用户自己熟悉的语言来开发应用程序。

对于 MATLAB 外部接口，有以下几个建议。

1. 选择与外部接口的方式

1）根据调用关系

MATLAB 外部接口可分为两种，第一种是其他应用程序调用 MATLAB 的函数或算法，另外一种是 MATLAB 调用其他语言的函数或算法。将本章内容按此分类：

第一种，外部程序调用 MATLAB 函数或算法：Excel、Java 和.NET 生成器、MATLAB 编译器、MATLAB 计算引擎、MATLAB COM 和 DDE 编程。

第二种，MATLAB 调用外部程序函数或算法：MEX 文件编程、MAT 文件、MATLAB COM 和 DDE 编程、MATLAB 硬件接口、MATLAB Web 服务。

MATLAB COM 和 DDE 编程可以使 MATLAB 作为服务器或客户端与其他应用程序建

立起服务器/客户端体系结构，从而实现软件之间的交互。MATLAB 作为服务器编程时，是外部程序调用 MATLAB 的函数或算法，而作为客户端编程时，是 MATLAB 调用外部程序的函数或算法。所以 MATLAB COM 和 DDE 编程同时属于上面两种分类。

若扩展 MATLAB 的功能，或用其他技术弥补 MATLAB 的不足之处，应该考虑使用 MATLAB 调用外部程序的函数或算法。这类应用有以下几方面。

（1）MEX 文件可以实现调用其他语言编写的程序或算法，来解决 MATLAB 解释执行效率低的问题，同时也可使用其他语言的现有算法等资源，而不必将其他语言编写的程序或算法用 M 文件重新编写。

（2）MATLAB Web 服务可使用 Internet 上的现有资源。

（3）使用 MAT 文件可实现 MATLAB 与其他的编程环境的数据交互。

（4）MATLAB 硬件接口实现了 MATLAB 与外围设备的交互，扩展了 MATLAB 的功能。

（5）MATLAB COM 编程使 MATLAB 可以直接使用现有的 COM 组件来简化应用程序开发，而 MATLAB DDE 编程可以与 Windows 应用程序进行数据交换，实现数据的交互。

若使用 MATLAB 的现有程序、算法和组件，或直接使用 MATLAB 的计算和绘图功能可以简化应用程序开发，则可以考虑使用外部程序调用 MATLAB 的函数或算法这类接口方式。这类应用有：

（1）Excel、Java 和.NET 生成器可以将 MATLAB 的函数或算法直接创建成组件，这些组件可以分别在 Excel 环境、Java 环境和.NET 程序中直接调用。若对组件打包后，还可以在没有安装 MATLAB 的环境中正常使用。

（2）MATLAB 编译器可以由 M 文件或混合 M 文件和 C 文件创建独立的应用程序或 C/C++共享库，创建后的应用程序或 C/C++库可以在没有安装 MATLAB 的环境中正常运行。

（3）外部程序调用 MATLAB 的计算和绘图功能可以使用 MATLAB 计算引擎，以简化程序开发。MATLAB 计算引擎在外部程序和 MATLAB 进程之间建立了一个桥梁，实现了二者的数据交互和命令传送，使得外部程序可以直接调用 MATLAB 的函数。

（4）MATLAB COM 和 DDE 编程实现了外部程序特别是 COM 组件和 MATLAB 的数据交互和命令传送。

2）根据应用类型

MATLAB 提供了诸多外部接口方式，各有其应用范围，如 Excel 生成器创建的组件只有在 Excel 程序中可以调用使用。Java 生成器创建的组件也只有在 Java 程序中才可以调用使用。考虑使用 MATLAB 外部接口方式时，应考虑所需的应用范围和应用类型。

（1）对于针对性较强的应用，如在 Excel、Java 和.NET 程序中调用 MATLAB 函数或算法，最好使用其对应的生成器。利用生成器创建和打包的组件可以脱离 MATLAB 环境使用。Excel 程序也可以使用 Excel Link 来调用 MATLAB 函数或算法，但此时 MATLAB 作为后台工作，而不能脱离 MATLAB 环境。

（2）若 C 和 Fortran 程序调用 MATLAB 的函数或算法，可以直接在程序中调用 MATLAB 计算引擎，但是调用 MATLAB 计算引擎的程序运行时不能脱离 MATLAB 环境。

（3）对于 C/C++程序，可以选择使用 MATLAB 编译器。使用 MATLAB 编译器可以开发独立运行的 C/C++的应用程序和共享库。创建的应用程序或库可以脱离 MATLAB 环境。

（4）MATLAB 编译器同时可以由 M 文件生成可独立运行的 C/C++应用程序，但不是

对于所有的 M 文件都可以转化。

（5）若调用现有组件，如 Windows 的日历和 mediaplayer 等，使用 COM 和 DDE 编程可简化软件开发。但是这种方法移植性不强，因为使用 COM 组件的 MATLAB 应用程序必须在安装此 COM 组件的机器上才能正常运行。

（6）访问连接计算机上串口的外围设备使用 MATLAB 的硬件接口，对于特定的设备，可能会有专门的命令，需要查询相应文档。

（7）访问网络上的服务使用 Web 服务，只是注意 Web 服务编程是不可靠的，因为 Web 服务是在不断变化的，现在可用的服务，以后可能不再免费或不再提供。

（8）对于 MATLAB 费时的循环等指令，可以将相应代码段写成等价 C 语言程序或 Fortran 语言程序，然后编译成 MEX 文件以解决 MATLAB 效率瓶颈问题。

（9）MATLAB 若调用现有的 C 和 Fortran 函数或算法，也可以考虑 MEX 文件方式。

（10）将数据导入或导出 MATLAB 可以使用 MAT 文件，另外与其他应用程序数据交互也可以使用 MAT 文件。

3）根据是否可以脱离 MATLAB 环境

按照是否可以脱离 MATLAB 环境，可以分为两种类型。如果要求应用程序的可移植性，即在没有安装 MATLAB 的机器上同样可以正常运行，则必须选择可以脱离 MATLAB 环境的接口方式。反之，如果选择必须有 MATLAB 环境支持的接口方式，则开发的应用程序必须在安装 MATLAB 的机器上才能正常运行。

（1）可以脱离 MATLAB 环境的接口方式。

利用 MATLAB 的编译器可以创建独立与 MATLAB 环境的应用程序和库。所以 MATLAB 编译器的相关应用和 MATLAB 编译器扩展和的接口方式都可以脱离 MATLAB 环境。这类接口方式有以下两种。

- MATLAB 编译器创建独立 C/C++应用程序和库。
- Excel、Java 和.NET 生成器创建和打包的组件。

（2）必须有 MATLAB 环境支持的接口方式。

和 MATLAB 编译器无关或关系不大的接口方式工作时，MATLAB 作为工作环境，或者 MATLAB 进程作为后台服务器，在后台完成必要的数据计算和绘图功能支持。这类接口方式有以下几种。

- MATLAB MEX 文件。
- MATLAB 的 MAT 文件接口。
- MATLAB 计算引擎。
- MATLAB Web 服务。
- MATLAB 硬件接口。
- MATLAB COM 和 DDE 编程。

2．编程语言方面

在编程方面，Visual Studio.NET 是比较流行的开发环境，集成了 Visual C++、Visual Basic、Visual J#、Visual C#等语言。使用 Visual Studio.NET 基本可以完成 MATLAB 所有的外部接口应用开发，如创建 MEX 文件、MAT 文件、Excel、Java 和.NET 生成器应用程序开发。同时，Visual Studio.NET 强大的图形程序调试功能可以方便程序开发。

另外，每个 MATLAB 外部接口也可根据需要单独选择编程语言。

（1）用到 Visual C++开发环境的比较多，如 MEX 编程、MAT 文件编程和 MATLAB 计算引擎编程。使用 Visual C++的开发环境还有利于程序调试。

（2）在开发.NET 生成器组件应用程序时，用户需安装 Visual Studio.NET 开发环境，2003 和 2005 版本都可以。

（3）开发 Excel 生成器组件应用程序时，由于要 VBA 编程环境，所以需要 Visual Basic 开发环境。

（4）开发 MEX 文件可以用 C 语言和 Fortran 语言。C 语言相对比较流行和简单，加上 Visual C++的图形综合开发环境更加增加了 C 语言的使用范围，用户比较多。但是 Fortran 语言在科学计算方面有其独特优势，读者可以根据具体情况选择。

（5）MATLAB 编译器、Excel 生成器、Java 生成器、.NET 生成器可用 mcc 命令或配置工具。

（6）MATLAB Web 服务、硬件接口、COM 编程、DDE 编程使用 MATLAB 语言。

在后面的章节中将对上述内容作更为详细的介绍。

10.2 MATLAB MEX 文件

通过 MEX 文件可以在 MATLAB 中像调用内嵌函数一样调用现有的使用 C 语言和 Fortran 等语言编写的函数，实现了代码重用，同时也能解决 MATLAB 循环效率低的缺点，提高 MATLAB 环境中数据处理的效率。

本节讲解 MEX 文件的系统配置，并以实例讲解 C 语言和 Visual C++环境中 MEX 文件的建立与调试方法。

10.2.1 MATLAB 的 MEX 文件

MEX 文件是按照一定的格式，使用 C 语言或 Fortran 语言编写，由 MATLAB 解释器自动调用并执行的动态链接库。MATLAB 7.1 版本（R14SP3）前，在 Windows 操作系统中，这种文件类型的后缀为.dll（dynamic link library），MATLAB 7.1 版本中，对 MEX 文件的后缀做了较大修改，其中 32 位的 Windows 操作系统中的 MEX 文件后缀从原来的“.dll”改成了“mexw32”。而且 MEX 文件的后缀是平台和操作系统相关的，如表 10-1 所示列出了常见的平台与对应的 MEX 文件后缀名。

表 10-1 不同平台中 MEX 文件的后缀名

平台(Platform)	MEX 文件后缀（MATLAB Extension）
Linux (32-bit)	mexglx
Linux x86-64	mexa64
Macintosh (PPC)	mexmac
Macintosh(Intel)	mexmaci
64-bit Solaris SPARC	mexs64
Windows (32 位)	mexw32
Windows x64	mexw64

从表 10-1 可以看出，现在的 MATLAB 版本在 32 位的 Windows 中的 MEX 文件后缀取代了“.dll”而使用“.mexw32”后缀。所以为了能使以前编译的 MEX 能正常工作，最好在安装了 MATLAB 7.1 及后续版本后对所有的 MEX 文件重新编译一遍，以免在新版本的 MATLAB 中出现难以察觉和调试的错误。另外在 MATLAB 7.0.4 版本中编译的后缀为“.dll”的 MEX 文件在 MATLAB 7.1 可以照常使用（即 MATLAB 7.0.4 中编译的 MEX 文件可以不重新编译）。

如果有两个同名的 MEX 文件，仅后缀不同，一个为“.mexw32”，另一个为“.dll”，如有 timestwo.dll 和 timestwo.mexw32 两个文件，那么有两种情况。

- 若这两个个文件在同一目录下，MATLAB 使用后缀为“.mexw32”的 MEX 文件。
- 若这两个文件不在同一个目录下，那么按照 MATLAB 搜索路径的顺序，MATLAB 使用先搜索到的路径的 MEX 文件。

MATLAB 7.1 及后续版本中，不用任何参数运行 mex 命令时，将会显示：

```
mex timestwo.c
```

会将 timestwo.c 默认地编译成 timestwo.mexw32 MEX 文件，并且此文件在 7.1 前的 MATLAB 中是不识别的。不过在 MATLAB 7.1 以及后续版本中也可以将 C 语言 MEX 文件编译成后缀为“.dll”的文件，而这些后缀为“.dll”的 MEX 文件可以为 MATLAB 7.1 前的版本所识别和正常使用，从而解决了不同版本之间的 MEX 文件的兼容问题。在 MATLAB 7.1 以及后续版本中输入：

```
mex timestwo.c -output timestwo.dll
```

会生成生成.dll 文件，因为“-output”选项显示地指定了目标文件的名字和后缀。

在创建 MEX 文件的目录中若已经有同名的 MEX 文件，而仅仅后缀不同，为了避免不经意的文件屏蔽，在创建 MEX 文件的同时，MATLAB 会为先存在的那个 MEX 文件自动改名，方法是给文件名添加后缀“.old”。如已经存在 timestwo.mexw32，执行下面命令：

```
mex timestwo.c -output timestwo.dll
```

创建成功后会有下面的警告：

```
Warning: Renaming "timestwo.mexw32" to "timestwo.mexw32.old" to avoid name conflicts.
```

调用 MEX 文件比较方便，直接在控制窗口输入 MEX 文件名即可，与调用 M 函数文件类似。如调用前面的 timestwo.mexw32 MEX 文件，只要输入：

```
timestwo(3)
```

则会显示结果：

```
ans =
     6
```

对于同名的 MEX 文件和 M 文件，MATLAB 优先调用 MEX 文件。

MATLAB 外部程序接口函数库提供了丰富的库函数，分为两类，第一类是 mx 为前缀，第二类是 mex 为前缀。这两类函数的功能以及使用范围有所不同。

- 前缀为 mx 的函数库的主要功能是为用户提供了与 C/C++语言或 Fortran 语言中创建、操作 MATLAB 的基本数据类型 mxArray 的方法。
- 前缀为 mex 的函数主要功能是 C/C++语言或 Fortran 语言与 MATLAB 环境的交互，而且只能在 MEX 文件中使用。

10.2.2 MEX文件系统设置

MEX文件的编写和编译需要两个基本条件：一是必须安装MATLAB应用程序接口组件和相关工具，二是要有C语言或Fortran语言的编译器。另外，如果是在Windows平台上，那么C语言的编译器还必须能够支持32位的Windows动态链接库。

当具备这两个条件后，还需要在此基础上对MATLAB系统进行设置，使MATLAB系统知道使用系统的哪一个C语言编译器，以及其参数和路径。MATLAB提供了相关的系统命令来完成这个过程。

本书中的配置是Microsoft XP（SP2）操作系统以及MATLAB R2007a版本，使用Microsoft Visual C++.NET 2005中的编译器。在MATLAB控制窗口输入：

```
mex -setup
```

将会显示：

```
Please choose your compiler for building external interface (MEX) files:
Would you like mex to locate installed compilers [y]/n?
```

若选择y，系统就会自动载入当前系统中安装的所有C语言编译器信息供用户选择。如选择y后，显示下面结果：

```
Select a compiler:
[1] Lcc-win32 C 2.4.1 in D:\MATLABR2007a\sys\lcc
[2] Microsoft Visual C++ 6.0 in D:\Program Files\Microsoft Visual Studio 6.0
[3] Microsoft Visual C++ .NET 2005 in D:\Program Files\Microsoft Visual Studio .NET 2005
[0] None
Compiler:
```

提示用户选择一个作为C语言的默认编译器，上面的显示结果决定于使用的计算机上安装的软件。其中选项[1]的Lcc是MATLAB中自带的C语言编译器。若输入3后按回车键，则显示下面结果：

```
Please verify your choices:

Compiler: Microsoft Visual C++ .NET 2005
Location: d:\Program Files\Microsoft Visual Studio .NET 2005

Are these correct?([y]/n):
```

让用户确认所作的选择，输入*y*确认选择，显示下面结果：

```
Trying to update options file: C:\Documents and Settings\Administrator\Application Data\MathWorks\
MATLABR2007a\mexopts.bat
From template:        D:\MATLABR2007a\bin\win32\mexopts\msvc71opts.bat

Done . . .
```

上述提示表明设置成功，至此，系统配置完毕。

10.2.3 C 语言 MEX 文件的建立

1．MEX 文件的结构

MEX 文件的代码由两个独立的部分组成，MEX 格式的 C 语言文件必须包含下面这两个部分。

- 计算子程序（Computational Routine），是完成实际计算功能的所有代码，可以是包含输入输出的数值计算。
- 入口子程序（Gateway Routine），是 MATLAB 和计算子程序的接口，在 MEX 文件中，入口子程序作为主函数调用计算子程序完成计算功能。

其中，计算子程序的名字可以任意取，而入口子函数必须是名为 mexFunction，并且输入参数数目和名称固定。入口函数 mexFuncton 的一般形式如下：

```
void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
    /* 用户特定的代码... */
}
```

其中 prhs 是包含输入参数的数组，整型 nrhs 表示输入参数的个数，plhs 为输出参数数组，整型 nlhs 是输出函数的个数。

在 MEX 文件中可以没有计算子程序，而必须有入口子程序，下面是一个 MEX 程序：

```
#include "mex.h"
void mexFunction(int nlhs, mxArray *plhs[],int nrhs,const mxArray *prhs[])
{
    mexPrintf("Hello MATLAB World!");
}
```

保存编译后，运行此程序显示下面结果：

```
Hello MATLAB World!>>
```

而若要完成实际计算功能，或者利用现有的 C 语言程序，则要利用计算子程序。下面以 MATLAB 自带的一个 C 语言 MEX 文件例子来说明 MEX 文件的结构。此例功能十分简单，将输入参数（必须是非复数标量-noncomplex scalar）乘以 2 后返回结果，名为 timestwo.c，在 matlabroot\extern\examples\refbook 目录（matlabroot 为本机 MATLAB 的安装根目录，本章以后内容同）下，这个例子简单、结构清晰、而且包含了 C 语言 MEX 文件的各个部分，能较好的说明 C 语言 MEX 文件的一般结构，其内容如下：

timestwo.c

```
#include "mex.h"     /* C 语言 MEX 文件的文件头必须有这一行 */

/* timestwo 函数是 C 语言函数，为计算子程序*/
void timestwo(double y[], double x[])
{
  y[0] = 2.0*x[0];
}

/*mexFucntion 是固定的入口子程序，参数也固定*/
void mexFunction( int nlhs, mxArray *plhs[],
```

```
                        int nrhs, const mxArray *prhs[] )
{
/* 创建必要临时变量*/
  double *x,*y;
  mwSize mrows,ncols;

  /*下面程序段检查“参数个数”是否合法 */
  if(nrhs!=1) {
    mexErrMsgTxt("One input required.");
  } else if(nlhs>1) {
    mexErrMsgTxt("Too many output arguments");
  }

  /*下面程序段检查“参数类型”是否合法 */
  mrows = mxGetM(prhs[0]);
  ncols = mxGetN(prhs[0]);
  if( !mxIsDouble(prhs[0]) || mxIsComplex(prhs[0]) ||
      !(mrows==1 && ncols==1) ) {
        mexErrMsgTxt("Input must be a noncomplex scalar double.");
  }

/* 为输出参数创建变量 */
  plhs[0] = mxCreateDoubleMatrix(mrows,ncols, mxREAL);

/* 为计算子程序的输入/输出参数赋值*/
  x = mxGetPr(prhs[0]);
  y = mxGetPr(plhs[0]);

  /* 调用 计算子程序 timestwo*/
  timestwo(y,x);
}
```

由此可以看出 C 语言 MEX 文件的基本结构如下：

```
#include "mex.h"   /* 必需的一行 */

/* 计算子程序，函数名、参数名称和参数个数任意 */
void Computational_Routine(a,b...)
{
    /* 进行实际的计算，完成此 MEX 文件的功能   */
}

/* 入口子程序，函数名、参数个数和参数名称必须和下面的严格一致 */
void mexFunction(
    int nlhs, mxArray *plhs[],
    int nrhs, const mxArray *prhs[])
{
    /*
    * 1: 检查输入参数的合法性
    * 2: 为输出参数创建变量
    * 3: 为计算子程序的输入/输出参数赋值（指针）
    * 4: 调用计算子程序完成计算
```

```
    */
    Computational_Routine(a,b..)
}
```

2．MEX 文件执行流程

C 语言的 MEX 文件中，入口程序是主函数，计算子程序是子函数，故首先调用入口子程序，然后检查其输入/输出参数，完成必要操作后，调用计算子程序。其中入口子函数中，可以访问 mxArray 结构体中的数据，并且可在 C 代码编写的计算子程序中对这些数据进行操作。如下面的表达式：

```
mxGetPr(prhs[0])
```

返回一个指向 mxArray 结构体中实数类型数据的 double *型的指针，此指针存储在 prhs[0]中。然后在 C 语言计算子函数中就可以将此指针当作 C 语言中普通的 double *型的指针进行操作了。

在入口子函数中调用完 C 语言计算子函数后，可将 mxArray 类型的指针指向返回的数据。这样 MATLAB 可以区分出计算子函数的返回数据和 MEX 文件的返回数据。

图 10-1 图示了 MEX 文件的入口子程序接收输入/输出参数并返回给 MATLAB。

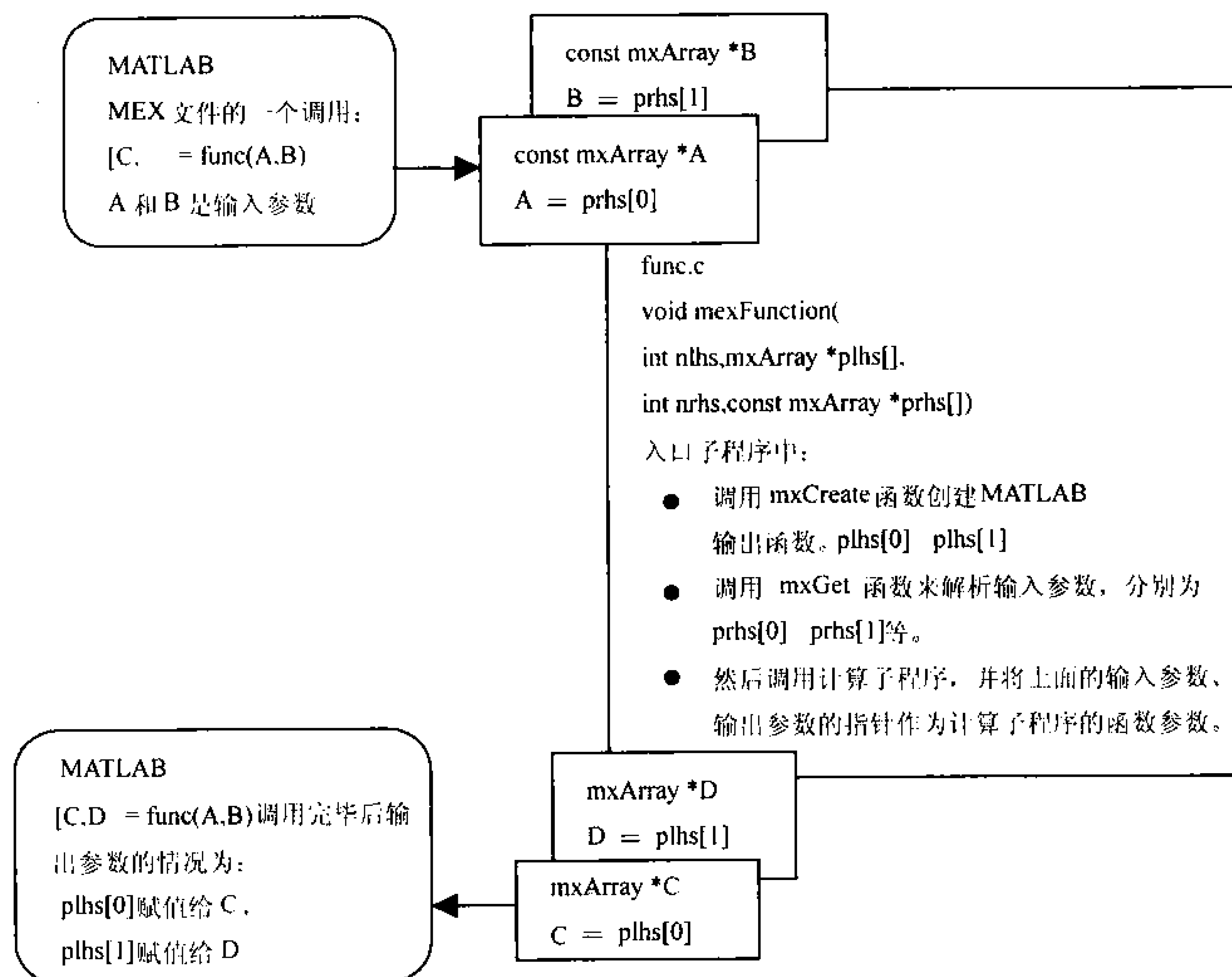


图 10-1　MEX 文件参数赋值情况

图 10-1 中的 MEX 文件的函数调用为[C, D] = func(A,B)，告诉 MATLAB 变量 A.、B 为 MEX 文件的输入参数，C 和 D 为输出参数。入口子程序 func.c 调用 mxCreate 函数为输出参数创建 MATLAB 数组，并将 plhs[0]、plhs [1]等指针指向这些新分配的 MATLAB 数组；调用 mxGet 函数解析传进来的输入参数 prhs[0]，prhs [1]等。最后，调用计算子程序，并将上面的输入/输出参数的指针作为计算子程序的函数参数。

MEX 文件的两个组成部分可以是独立的，也可以是组合在一起的，但是在这两种情况下，文件都必须包含 mex.h 头文件，即文件头必须有“#include "mex.h"”语句。这样入口子程序和计算子程序才能正确的声明。

参数 nlhs 和 nrhs 包含了 MEX 文件输入参数和输出参数的个数，在 MATLAB 语言的语法中，函数有以下一般形式：

```
[a,b,c,...] = fun(d,e,f,...)
```

其中省略号（...）表示可以有任意个此类参数。参数 plhs 和 prhs 都是向量，包含了 MEX 文件参数的指针。注意 plhs 和 prhs 都声明成了 Array *类型，所以这两个变量都指向 MATLAB 的数组。prhs 是一个长度为 nrhs 的指针数组，指向 MEX 文件的输入参数，而 plhs 为长度是 nlhs 的指针数组，指向函数产生的输出参数。

如在 MATLAB 控制窗口输入命令调用 MEX 文件，显示如下：

```
x = fun(y,z);
```

MATLAB 的解释器调用 mexFunction 函数，其参数情况如图 10-2 所示。

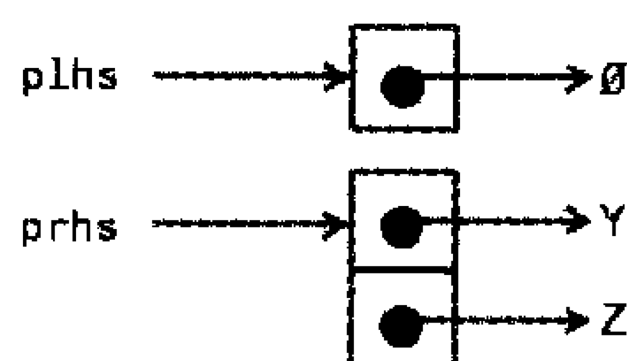


图 10-2　mexFunction 参数情况

其中 plhs 是一个有一个元素的 C 语言数组，这个元素是空指针，prhs 是两个元素的 C 语言指针数组，第一个元素指向 mxArray 数组 Y，另外一个指向 mxArray 数组 Z。参数 plhs 没有指向任何值是因为输出参数 *x* 在计算子程序调用前还没有创建。入口子程序的任务是创建一个输出参数数组然后将 plhs[0]指向这个数组。

3．创建 C 语言 MEX 文件

MATLAB 的 API 函数提供了各种函数来处理 MATLAB 所支持的各种数据类型。每个数据类型都有相对应的一类函数。本节先介绍一个操作实数类型变量的简单例子。然后讨论对各种数据类型如何传递输入参数、进行操作然后返回操作结果，以及如何处理多个输入和输出。

1）简单例子

例 10.1　编写 MEX 文件，以一个标量为输入，乘以 2 后返回结果。

这个例子是 MATLAB 自带的，源文件 timestwo.c 的文件结构已经在“MEX 文件结构”一节中分析过了。下面是 MEX 文件的编写方法：

timestwo.c

```
#include "mex.h"
void timestwo(double y[], double x[])
{
  y[0] = 2.0*x[0];    /* 计算子程序，完成实际计算：y = 2.0×x   */
}

/* 编辑 mexFucntion 入口子程序，格式是固定的   */
void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[] )
{
  double *x,*y;              /* 为输入/输出参数创建变量，类型为指针   */
  mwSize mrows,ncols;        /* 输入参数的行数和列数（MATLAB 的数据类型默认为矩阵）*/
```

```
/* 检查输入/输出参数个合法性 */
if(nrhs!=1) {
   /*必须一个输入参数 */
   mexErrMsgTxt("One input required.");
} else if(nlhs>1) {
   //不能返回多于一个输出参数
   mexErrMsgTxt("Too many output arguments");
}

/* 检查输入参数类型是否合法，输入参数必须是非复数标量 */
/* 获取输入参数的行数 */
mrows = mxGetM(prhs[0]);
/* 获取输入参数的行数列数 */
  ncols = mxGetN(prhs[0]);
  if( !mxIsDouble(prhs[0]) || mxIsComplex(prhs[0]) ||
       !(mrows==1 && ncols==1) ) {
     /* 输入参数若不是 double 型，或是复数，或不是标量则报错 */
     mexErrMsgTxt("Input must be a noncomplex scalar double.");
  }

  /* 为输出参数创建变量 */
  plhs[0] = mxCreateDoubleMatrix(mrows,ncols, mxREAL);

  /*为计算子程序的输入参数赋值，值为指向入口子程序参数 prhs[0]的指针 */
  x = mxGetPr(prhs[0]);
  /* 为计算子程序的输出参数赋值，值为指向刚创建的输出参数变量 plhs[0]的指针 */
  y = mxGetPr(plhs[0]);

  /* 设置好输入/输出参数后，就可以调用计算子程序 timestwo 了*/
  /* 执行 y = 2*x 操作，并返回结果 y 的值 */
  timestwo(y,x);
}
```

在 C 语言的程序中，函数参数的检查是在编译阶段完成的，在 MATLAB 环境中，可以为 M 函数传递任何数目的参数，在 MEX 文件中同样如此。所以在程序中必须考虑周全，能处理任意个数、任意支持类型的输入和输出参数的情况。

编译链接上面 C 语言的 MEX 文件源程序，在 MATLAB 的控制窗口中输入：

```
mex timestwo.c
```

上述命令在 Windows 环境下会创建一个名为 timestwo.mexw32 的 MEX 文件，后缀可能会由此命令所运行的平台不同而不同，具体细节见 10.2.1 节中的表 10-1，可以像调用 M 函数一样调用 timestwo：

```
x = 2;
y = timestwo(x)
y =
     4
```

若在 MATLAB 控制窗口中使用 mex 命令，则 MATLAB 会调用名为 mex.m 的 M 脚本文件来完成编译工作。另外，操作系统提供了编译工具，在 Windows 中是 mex.bat，在 UNIX 中是 mex.sh，所以在操作系统的命令行中输入：

mex filename

也会得到一个编译的 MEX 文件。比如在 Windows 的 cmd 窗口中可以像在 MATLAB 环境中一样调用 mex 命令，格式和参数都一致。

例 10.2 用 mxGetScalar 函数检查输入参数合法性，重新编写例 10.1。

在例 10.1 中，标量被看做 1×1 的矩阵，所以在判断输入参数的合法性时，就可以利用专门的矩阵 API 函数 mxGetScalar，此 API 函数直接返回标量的值，而不是标量副本的指针。

源文件为 timestwoalt.c，与 timestwo.c 在同一目录下，内容如下：

timestwoalt.c

```
#include "mex.h"

void timestwo_alt(double *y, double x)
{
  *y = 2.0*x;
}

void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[] )
{
  double *y;
  double  x;

 /* 检查参数 */
  if (nrhs != 1) {
    mexErrMsgTxt("One input argument required.");
  } else if (nlhs > 1) {
    mexErrMsgTxt("Too many output arguments.");
  } else if (!mxIsNumeric(prhs[0])) {
    mexErrMsgTxt("Argument must be numeric.");
  } else if (mxGetNumberOfElements(prhs[0]) != 1 || mxIsComplex(prhs[0])) {
    mexErrMsgTxt("Argument must be non-complex scalar.");
  }
  /* 为输出参数创建变量 */
  plhs[0] = mxCreateDoubleMatrix(1,1,mxREAL);

  /*
    为参数 x、y 赋值，x 为值，而 y 为指针
    （由于 MATLAB 没有值传递，所以用指针才能得到修改后的 y 值，
    不然修改的是 y 的一个副本，为临时变量，在函数返回时，y 值没有变化，
    不能得到希望的结果）
*/
x = mxGetScalar(prhs[0]);
y = mxGetPr(plhs[0]);

/* 调用 timestwo_alt 子函数 */
timestwo_alt(y,x);
}
```

例 10.3 编写 MEX 文件，实现实数加法。

分析了 MATLAB 所带的例子 timestwo.c 后，本节参照此例来创建一个 MEX 文件。完成实数的加法计算的 C 语言程序如下：

```
#include <math.h>
double double_add(double x, double y)
{
  return x + y;
}
```

由此，需要编写 MEX 格式的 C 语言程序 add.c，然后再编译成 MEX 文件。

第一步：编写计算子程序，内容如下：

```
void double_add(double *z, double x, double y)
{
  *z = x + y;
}
```

其中参数类型参照 timestwoalt.c 文件，将实数标量看成 1×1 的矩阵，可以利用专门的矩阵 API 函数来优化函数。

第二步：编写 mexFunction 入口子程序，由于格式固定，也便于编写，笔者编写的 mexFunction 如下：

```
void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[] )
{
  /* 创建临时变量 */
  double *z;
  double   x;
  double   y;

 /* 检查参数 */
  if (nrhs != 2) {
    mexErrMsgTxt("Two input argument required.");
  } else if (nlhs > 1) {
    mexErrMsgTxt("Too many output arguments.");
  } else if (!mxIsNumeric(prhs[0]) || !mxIsNumeric(prhs[1])) {
    /* 需要检查两个输入参数 x 和 y 是否为数值型   */
    mexErrMsgTxt("Both The Two Arguments must be numeric.");
  } else if (mxGetNumberOfElements(prhs[0]) != 1 || mxIsComplex(prhs[0]) ||
          mxGetNumberOfElements(prhs[1]) != 1 || mxIsComplex(prhs[1])) {
/* 需要检查两个输入参数 x 和 y 是否为标量 */
    mexErrMsgTxt("Argument must be non-complex scalar.");
  }
  /* 为输出参数创建变量 */
  plhs[0] = mxCreateDoubleMatrix(1,1,mxREAL);

  x = mxGetScalar(prhs[0]);
  y = mxGetScalar(prhs[1]);
  z = mxGetPr(plhs[0]);

  /* 调用 double_add 子函数 */
  double_add(z,x,y);
```

```
}
```

在 MATLAB 的控制窗口中输入：

```
mex double_add.c
```

则会生成 double_add.mexw32 MEX 文件，在控制窗口中调用 double_add 测试函数是否完成预定功能，显示如下：

```
double_add(pi,6)
```

得出的结果如下：

```
ans =
    9.1416
```

可见 double_add.c 编译成功，而成功地完成了预计功能。若编译不成功，MATLAB 会给出出错的行数和对应的错误编号，以及具体的出错信息。

另外可以看到，mexFunction 是没返回值的，它不是通过返回值把结果传回 MATLAB，而是通过对参数 plhs 的赋值。再调用下面语句：

```
result = double_add(pi,6)
```

此时 mexFunction 4 个参数的值如下。

- nlhs = 1，说明 mexFunction 函数有一个输出变量，即 result。
- nrhs = 2，说明 mexFunction 函数有两个输入变量，即 pi 和 6。
- plhs 是一个数组，其元素的类型为指针，该指针指向的数据的数据类型为 mxArray。此时 mexFunction 函数仅有一个输出变量，所以该数组只有一个指针，plhs[0]指向的值会赋值给 result。
- prhs 和 plhs 类似，不同的是其元素指向输入变量，此时有两个输入变量，则该数组有两个元素 prhs[0]和 prhs[1]，prhs[0]指向了 pi，prhs[1]指向了 6。另外 prhs 与 plhs 的类型不同，prhs 是 const 型的指针数组，其指向数据的内容不能修改，但是可以改变指针的指向，具体细节参照 C 语言文献中关于 const 关键字的说明。

为什么 plhs 和 prhs 都是指向 mxArray 类型的指针数组呢？首先，MATLAB 允许有任意个输入/输出参数，所以 plhs 和 prhs 都要是数组。其次，MATLAB 没有参数传递没有地址传递方式，所以要想返回修改后的结果，必须采用 C 语言的指针，故 plhs 和 prhs 都是指针数组。最后，MATLAB 的基本数据类型是矩阵，即 array 类型，如 MATALB 中有 double array、cell array 和 struct array 等。而在 C 语言中，MATLAB 的 array 使用 mxArray 类型来表示。在 MATLAB 中常数 pi 和 6 也被看做是 array，只不过是 1×1 的 double array。所以 plhs 和 prhs 是指向 mxArray 类型的指针数组。

从上面的简单例子中可以看出，MEX 文件实现了一种 C 语言与 MATLAB 的接口，其实际的计算功能仍在 C 语言形式的计算子程序中完成，而入口子程序的功能是检查参数以匹配 C 语言的参数规范，比如 C 语言中只能有一个返回结果，而 MATLAB 可以多个，这样，在 MATLAB 中以多个返回值调用 MEX 文件时，就要对参数个数进行匹配。然后将设置、匹配好的参数赋值给计算子程序，计算子程序根据参数完成计算功能，最后将计算结果返回给 MATLAB。

所以有 C 语言编写的大型程序时，不必用 MATLAB 语言重新编写，只要将此 C 语言程序作为一个计算子程序，然后编写一个入口子程序，完成参数的匹配，然后编译成 MEX 文件即可。MEX 文件的另外一个功能是可以将 MATLAB 编程中的瓶颈问题，如多重循环

等，将此类费时的指令用 C 语言实现，然后作必要的入口子程序，编译成 MEX 文件，可以有效地提高 MATLAB 的效率。

2）不同数据类型的参数传递

由于 MATLAB 和 C 语言的数据类型有很大不同，在上面的例子中可以看到，在入口函数 mexFunction 中的参数操作中，有个必要的步骤是将 MATLAB 的输入/输出参数传递给 C 语言的计算子程序以完成计算，所以必须对二者的数据类型进行匹配。

下面分别讨论如何编写关于不同数据类型操作的 MEX 文件。所用的例子都是 MATLAB 自带的例子文件，和 timestwo.c 在同一个目录下，另外，为了节省篇幅，去掉了英文注释和若干空行，中文为笔者添加。

（1）字符串（Strings）

例 10.4 编写 MEX 文件，实现将一个字符串反转的功能。

这个例子很好的演示了 MATLAB 的字符串与 C 语言字符串之间的相互转换以及其他对字符串的相关操作。源文件为 revord.c，其内容如下：

revord.c

```
#include "mex.h"

void revord(char *input_buf, mwSize buflen, char *output_buf)
{
  mwSize i;

  if (buflen == 0) return;

  /* 反转输入字符串 */
  for(i=0;i<buflen-1;i++)
    *(output_buf+i) = *(input_buf+buflen-i-2);
}

void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[])
{
    char *input_buf, *output_buf;
    mwSize buflen;

    /* 检查参数个数的合法性 */
    if(nrhs!=1)
      mexErrMsgTxt("One input required.");
    else if(nlhs > 1)
      mexErrMsgTxt("Too many output arguments.");

    /* 输入参数必须为字符串*/
    if ( mxIsChar(prhs[0]) != 1)
      mexErrMsgTxt("Input must be a string.");

    /* 输入参数必须为行向量 */
    if (mxGetM(prhs[0])!=1)
```

```
        mexErrMsgTxt("Input must be a row vector.");

    /* 获取输入字符串的长度 */
    buflen = (mxGetM(prhs[0]) * mxGetN(prhs[0])) + 1;

    /* 为输出字符串分配内存 */
    output_buf=mxCalloc(buflen, sizeof(char));

    /* 将输入的 MATLAB 类型的字符串转换为 C 语言类型的字符串变量，
       并赋值给 input_ buf */
    input_buf = mxArrayToString(prhs[0]);

    if(input_buf == NULL)
        mexErrMsgTxt("Could not convert input to string.");

    /* 调用子程序 */
    revord(input_buf, buflen, output_buf);

    /* 最后将 C 语言类型的字符串 output_buf 转换为 MATLAB 类型的字符串，并赋值给
mexFunction 的输出参数 plhs*/
    plhs[0] = mxCreateString(output_buf);
    mxFree(input_buf);
    return;
}
```

C 语言中，标准的动态分配内存函数为 calloc，而此例中，API 函数 mxCalloc 取代了 calloc 函数。其中 mxCalloc 使用 MATLAB 的内存管理器来分配内存并初始化为 0。

在任何需要使用 C 语言的 calloc 的场合中，都必须使用 mxCalloc 来替代。另外，mxMalloc 函数和 mxRealloc 函数也类似：在需要使用 C 语言的 malloc 时，必须使用 mxMalloc 函数来替代，在需要使用 C 语言的 realloc 时，必须使用 mxRealloc 函数来替代。

编译此文件后，在控制窗口中输入：

```
x = 'hello world';
y = revord(x)
```

会得出以下结果：

```
The string to convert is 'hello world'.
y =
   dlrow olleh
```

（2）结构体和细胞数组

例 10.5　以 MEX 文件 phonebook.c 演示 C 语言中操作结构体和细胞数组。

如果结构体和细胞矩阵的数据为 mxArray 类型，则可以像其他数据类型一样传递结构体和细胞矩阵。但是实际的应用中这种情况极为少见，所以对于结构体必须用 mxGetField API 函数、对于细胞数组使用 mxGetCell API 函数来获得 mxArray 类型的指针。获取 mxArray 类型的指针后，就可以像其他类型的指针一样操作了，但是若将数据传递给 C 计算子程序的话，需要使用 API 函数，如 mxGetData 函数来访数据。

此例接收一个 $M \times N$ 的结构体矩阵，然后返回 1×1 的结构体，包含下列两种域。

- 字符串格式的输入参数会生成一个 $M \times N$ 的细胞数组。
- 数值型（非复数，标量）格式的输入参数生成一个 $M \times N$ 的数值数组。

phonebook.c

```
#include "mex.h"
#include "string.h"

#define MAXCHARS 80    /* 每个域的字符串的最大长度 */

void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[] )
{
    const char **fnames;         /* 域名的指针 */
    const mwSize *dims;
    mxArray     *tmp, *fout;
    char        *pdata=NULL;
    int         ifield, nfields;
    mxClassID   *classIDflags;
    mwIndex     jstruct;
    mwSize      NStructElems;
    mwSize      ndim;

    /* 检查输入/输出参数 */
    if(nrhs!=1)
      mexErrMsgTxt("One input required.");
    else if(nlhs > 1)
      mexErrMsgTxt("Too many output arguments.");
    else if(!mxIsStruct(prhs[0]))
      mexErrMsgTxt("Input must be a structure.");
    /* 获取输入参数 */
    nfields = mxGetNumberOfFields(prhs[0]);
    NStructElems = mxGetNumberOfElements(prhs[0]);
    /* 为存储 classIDflags 分配内存 */
    classIDflags = mxCalloc(nfields, sizeof(mxClassID));

    /* 检查空域、数据类型以及数据类型一致性，获取每个域的 classID */
    for(ifield=0; ifield<nfields; ifield++) {
    for(jstruct = 0; jstruct < NStructElems; jstruct++) {
        tmp = mxGetFieldByNumber(prhs[0], jstruct, ifield);
        if(tmp == NULL) {
        mexPrintf("%s%d\t%s%d\n", "FIELD: ", ifield+1, "STRUCT INDEX :", jstruct+1);
        mexErrMsgTxt("Above field is empty!");
        }
        if(jstruct==0) {
        if( (!mxIsChar(tmp) && !mxIsNumeric(tmp)) || mxIsSparse(tmp)) {
            mexPrintf("%s%d\t%s%d\n", "FIELD: ", ifield+1, "STRUCT INDEX :", jstruct+1);
            mexErrMsgTxt("Above field must have either string or numeric non-sparse data.");
        }
        classIDflags[ifield]=mxGetClassID(tmp);
```

```
            } else {
            if (mxGetClassID(tmp) != classIDflags[ifield]) {
                mexPrintf("%s%d\t%s%d\n", "FIELD: ", ifield+1, "STRUCT INDEX :", jstruct+1);
                mexErrMsgTxt("Inconsistent data type in above field!");
            } else if(!mxIsChar(tmp) &&
                  ((mxIsComplex(tmp) || mxGetNumberOfElements(tmp)!=1))){
                mexPrintf("%s%d\t%s%d\n", "FIELD: ", ifield+1, "STRUCT INDEX :", jstruct+1);
                mexErrMsgTxt("Numeric data in above field must be scalar and noncomplex!");
            }
            }
        }
        }

    /* 为存储指针分配内存 */
    fnames = mxCalloc(nfields, sizeof(*fnames));
    /* 获取域名的指针 */
    for (ifield=0; ifield< nfields; ifield++){
    fnames[ifield] = mxGetFieldNameByNumber(prhs[0],ifield);
    }
    /* 为输出参数创建一个 1×1 的结构体矩阵 */
    plhs[0] = mxCreateStructMatrix(1, 1, nfields, fnames);
    mxFree((void *)fnames);
    ndim = mxGetNumberOfDimensions(prhs[0]);
    dims = mxGetDimensions(prhs[0]);
    for(ifield=0; ifield<nfields; ifield++) {
    /* 创建细胞矩阵或数值数组 */
    if(classIDflags[ifield] == mxCHAR_CLASS) {
        fout = mxCreateCellArray(ndim, dims);
    }else {
        fout = mxCreateNumericArray(ndim, dims, classIDflags[ifield], mxREAL);
        pdata = mxGetData(fout);
    }
    /* 从输入参数中复制数据 */
    for (jstruct=0; jstruct<NStructElems; jstruct++) {
        tmp = mxGetFieldByNumber(prhs[0],jstruct,ifield);
        if( mxIsChar(tmp)) {
          mxSetCell(fout, jstruct, mxDuplicateArray(tmp));
        }else {
          mwSize      sizebuf;
          sizebuf = mxGetElementSize(tmp);
          memcpy(pdata, mxGetData(tmp), sizebuf);
          pdata += sizebuf;
        }
    }
    /* 为输出参数设置每个域的值 */
    mxSetFieldByNumber(plhs[0], 0, ifield, fout);
    }
    mxFree(classIDflags);
    return;
}
```

编译 phonebook.c 文件后，输入以下命令查看此 MEX 文件的功能：

```
friends(1).name = 'Jordan Robert';
friends(1).phone = 3386;
friends(2).name = 'Mary Smith';
friends(2).phone = 3912;
friends(3).name = 'Stacy Flora';
friends(3).phone = 3238;
friends(4).name = 'Harry Alpert';
friends(4).phone = 3077;
```

则 friends 为有 4 个元素的数组，每个元素为一个结构体，有两个域：phone 和 name。调用 phonebook MEX 文件：

```
phonebook(friends)
```

得到：

```
ans =
    name: {'Jordan Robert'   'Mary Smith'   'Stacy Flora'   'Harry Alpert'}
   phone: [3386 3912 3238 3077]
```

（3）复数

MATLAB 中的复数分成实部和虚部两部分来分开存储。MATLAB 提供了两个专门的 API 函数 mxGetPr 和 mxGetPi，这个两个函数分别返回指向一个复数的实部和虚部的指针，指针都为 double *类型。

例 10.6 编写 MEX 文件，实现计算两个复数的卷积（Convolve）的功能。

源文件为 convec.c，其内容如下：

convec.c

```
#include "mex.h"

void convec( double *xr, double *xi, mwSize nx,
             double *yr, double *yi, mwSize ny,
             double *zr, double *zi)
{
    mwSize i,j;

    zr[0]=0.0;
    zi[0]=0.0;
    /* 计算复数数组的卷积 */
    for(i=0; i<nx; i++) {
    for(j=0; j<ny; j++) {
        *(zr+i+j) = *(zr+i+j) + *(xr+i) * *(yr+j) - *(xi+i) * *(yi+j);
        *(zi+i+j) = *(zi+i+j) + *(xr+i) * *(yi+j) + *(xi+i) * *(yr+j);
    }
    }
}

void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[] )
{
    double  *xr, *xi, *yr, *yi, *zr, *zi;
```

```
    mwSize   rows, cols, nx, ny;

    /* 检查参数 */
    if(nrhs != 2)
      mexErrMsgTxt("Two inputs required.");
    if(nlhs > 1)
      mexErrMsgTxt("Too many output arguments.");
    /* 检查两个输入参数是否是行数组*/
    if( mxGetM(prhs[0]) != 1 || mxGetM(prhs[1]) != 1 )
      mexErrMsgTxt("Both inputs must be row vectors.");
    rows = 1;
    /* 检查输入参数是否为复数 */
    if( !mxIsComplex(prhs[0]) || !mxIsComplex(prhs[1]) )
      mexErrMsgTxt("Inputs must be complex.\n");

    /* 分别获取两个每个输入参数的长度 */
    nx = mxGetN(prhs[0]);
    ny = mxGetN(prhs[1]);

    /* 获取两个输入参数的实部和虚部的指针 */
    xr = mxGetPr(prhs[0]);
    xi = mxGetPi(prhs[0]);
    yr = mxGetPr(prhs[1]);
    yi = mxGetPi(prhs[1]);

    /* 为输出参数创建数组并将输出参数指针指向这个数组 */
    cols = nx + ny - 1;
    plhs[0] = mxCreateDoubleMatrix(rows, cols, mxCOMPLEX);
    zr = mxGetPr(plhs[0]);
    zi = mxGetPi(plhs[0]);

    /* 调用 C 计算子程序 */
    convec(xr, xi, nx, yr, yi, ny, zr, zi);

    return;
}
```

用 mex 命令对此文件成功编译后，在 MATLAB 的控制窗口中输入下列复数：

```
x = [3.000 - 1.000i, 4.000 + 2.000i, 7.000 - 3.000i];
y = [8.000 - 6.000i, 12.000 + 16.000i, 40.000 - 42.000i];
```

然后调用生成的 MEX 文件：

```
z = convec(x,y)
```

得到的结果如下：

```
z =
  1.0e+002 *
  0.1800 - 0.2600i   0.9600 + 0.2800i   1.3200 - 1.4400i   3.7600 - 0.1200i   1.5400 - 4.1400i
```

以上结果和 MATLAB 的专门计算卷积的内嵌函数 conv.m 生成的结果一致。

（4）稀疏矩阵

MATLAB 提供了一套可以在 MEX 文件中创建和操作稀疏矩阵的 API 函数。这些 API

函数可以访问和操作稀疏矩阵相关的两个重要参数 ir 和 jc。

例 10.7 编写 MEX 文件，演示操作稀疏矩阵。

源文件为 fulltosparse.c，内容如下：

fulltosparse.c

```
#include <math.h>
#include "mex.h"

/* 若使用的编译器中 NaN 等同于 0，那么编译此文件时，必须按照下面的格式
 *       mex -DNAN_EQUALS_ZERO fulltosparse.c
*/

#if defined(NAN_EQUALS_ZERO)
#define IsNonZero(d) ((d)!=0.0 || mxIsNaN(d))
#else
#define IsNonZero(d) ((d)!=0.0)
#endif

void mexFunction(
        int nlhs,          mxArray *plhs[],
        int nrhs, const mxArray *prhs[]
        )
{
    /* 声明变量 */
    mwSize m,n;
    mwSize nzmax;
    mwIndex *irs,*jcs,j,k;
    int cmplx,isfull;
    double *pr,*pi,*si,*sr;
    double percent_sparse;

    /*检查输入/输出参数的个数是否合法 */
    if (nrhs != 1) {
    mexErrMsgTxt("One input argument required.");
    }
    if(nlhs > 1){
    mexErrMsgTxt("Too many output arguments.");
    }

    /* 检查输入参数的数据类型  */
    if (!(mxIsDouble(prhs[0]))){
    mexErrMsgTxt("Input argument must be of type double.");
    }

    if (mxGetNumberOfDimensions(prhs[0]) != 2){
    mexErrMsgTxt("Input argument must be two dimensional\n");
    }

    /* 获取输入数据的大小和指针 */
```

```
m    = mxGetM(prhs[0]);
n    = mxGetN(prhs[0]);
pr = mxGetPr(prhs[0]);
pi = mxGetPi(prhs[0]);
cmplx = (pi==NULL ? 0 : 1);

/* 为稀疏矩阵分配内存   */
percent_sparse = 0.2;
nzmax=(mwSize)ceil((double)m*(double)n*percent_sparse);

plhs[0] = mxCreateSparse(m,n,nzmax,cmplx);
sr    = mxGetPr(plhs[0]);
si    = mxGetPi(plhs[0]);
irs = mxGetIr(plhs[0]);
jcs = mxGetJc(plhs[0]);

/* 复制非零值 */
k = 0;
isfull=0;
for (j=0; (j<n); j++) {
  mwSize i;
  jcs[j] = k;
  for (i=0; (i<m ); i++) {
if (IsNonZero(pr[i]) || (cmplx && IsNonZero(pi[i]))) {

  /* 检查非零元素是否适合分配的输出数组，否则，percent_sparse 增加 10%，
  * 重新计算 nzmax，然后扩大疏矩阵
   */
  if (k>=nzmax){
    mwSize oldnzmax = nzmax;
    percent_sparse += 0.1;
    nzmax = (mwSize)ceil((double)m*(double)n*percent_sparse);

    /* 确保 nzmax 至少增加 1 */
    if (oldnzmax == nzmax)
      nzmax++;

    mxSetNzmax(plhs[0], nzmax);
    mxSetPr(plhs[0], mxRealloc(sr, nzmax*sizeof(double)));
    if(si != NULL)
      mxSetPi(plhs[0], mxRealloc(si, nzmax*sizeof(double)));
    mxSetIr(plhs[0], mxRealloc(irs, nzmax*sizeof(mwIndex)));

    sr    = mxGetPr(plhs[0]);
    si    = mxGetPi(plhs[0]);
    irs = mxGetIr(plhs[0]);
  }
  sr[k] = pr[i];
  if (cmplx){
    si[k]=pi[i];
```

```
            }
            irs[k] = i;
            k++;
              }
            }
            pr += m;
            pi += m;
        }
        jcs[n] = k;
    }
```

用 mex 命令编译此文件后，在 MATLAB 的控制窗口输入：

```
full = eye(5)
full =
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

来创建一个 5×5 的全（full）矩阵，然后调用 fulltosparse MEX 文件来将全矩阵转换成稀疏矩阵。

```
spar = fulltosparse(full)
spar =
   (1,1)        1
   (2,2)        1
   (3,3)        1
   (4,4)        1
   (5,5)        1
```

（5）8、16 和 32 位数据

MATLAB 也为创建和操作有符号和无符号的 8、16 和 32 位数据提供了专门的 API 函数。这些 API 函数提供了支持这些数据类型的操作函数集，利用这些 API 函数，用户就在自己的 MEX 文件中对这些数据进行操作了。

其中 API 函数 mxCreateNumericArray 创建一个特定数据大小的 *N* 维数值数组。当创建了 MATLAB 的数组后，可以调用 mxGetData 和 mxGetImagData 来访问这些数据。这两个函数分别返回指向实部和虚部的指针。

例 10.8 编写 MEX 文件，创建一个 2×2 的矩阵，其元素的数据类型为无符号的 16 位整数，然后对这个矩阵每一个元素乘以 2，最后返回原矩阵和结果矩阵。

源文件为 doubleelement.c，其内容如下：

doubleelement.c

```
#include <string.h> /* C 语言的 memcpy()函数需要 string 文件头 */
#include "mex.h"

#define NDIMS 2
#define TOTAL_ELEMENTS 4

void dbl_elem(unsigned short *x)
```

```
{
   unsigned short scalar=2;
   int i,j;

   for(i=0;i<2;i++) {
     for(j=0;j<2;j++) {
       *(x+i*2+j) = scalar * *(x+i*2+j);
     }
   }
}

void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[] )
{
   const mwSize dims[]={2,2};
   unsigned char *start_of_pr;
   unsigned short data[]={1,2,3,4};
   size_t bytes_to_copy;

   (void) nlhs; (void) nrhs; (void) prhs;

   /* 调用计算子程序 */
   dbl_elem(data);

   /* 创建2×2 矩阵，元素为无符号的 16 位整数 */
   plhs[0] = mxCreateNumericArray(NDIMS,dims,mxUINT16_CLASS,mxREAL);

   start_of_pr = (unsigned char *)mxGetData(plhs[0]);
   bytes_to_copy = TOTAL_ELEMENTS * (size_t)mxGetElementSize(plhs[0]);
   memcpy(start_of_pr,data,bytes_to_copy);
}
```

用 mex 编译上述 MEX 文件后，在 MATLAB 控制窗口中输入：

```
doubleelement
```

得到的结果如下：

```
ans =
     2     6
     8     4
```

此函数的输出一个元素为无符号 16 位整数的 2×2 的矩阵。

（6）多维数值数组

利用 mxGetData 和 mxGetImagData 两个 API 函数可以操作多维数值数组，这两个函数分别返回指向多维数组中数据的实部和虚部的指针。

例 10.9 编写 MEX 文件，以 *N* 维 double 型数组为输入，然后返回此数组中非零值的索引。

源文件为 findnz.c，其内容如下。注意，MATLAB 中的索引值从 1 开始，而 C 语言从 0 开始。

findnz.c

```
#include "mex.h"

/* 若使用的编译器中 NaN 等同于 0，则必须按照下面的格式编译此文件：
```

```
*          mex -DNAN_EQUALS_ZERO findnz.c
*/

#if NAN_EQUALS_ZERO
#define IsNonZero(d) ((d)!=0.0 || mxIsNaN(d))
#else
#define IsNonZero(d) ((d)!=0.0)
#endif

void mexFunction(int nlhs, mxArray *plhs[],
            int nrhs, const mxArray *prhs[])
{
    /* 变量声明 */
    mwSize elements,j,cmplx;
    mwSize number_of_dims;
    mwSize nnz=0, count=0;
    double *pr, *pi, *pind;
    const mwSize *dim_array;

    /* 检查输入/输出参数的个数是否合法 */
    if (nrhs != 1) {
    mexErrMsgTxt("One input argument required.");
    }
    if (nlhs > 1){
    mexErrMsgTxt("Too many output arguments.");
    }

    /* 检查输入参数的数据类型 */
    if (!(mxIsDouble(prhs[0]))) {
      mexErrMsgTxt("Input array must be of type double.");
    }

    /* 获取输入参数的元素个数 */
    elements=mxGetNumberOfElements(prhs[0]);

    /* 获取数据 */
    pr=(double *)mxGetPr(prhs[0]);
    pi=(double *)mxGetPi(prhs[0]);
    cmplx = ((pi==NULL) ? 0 : 1);

    /* 为非零值计数，以便为输出参数分配合适大小的内存 */
    for(j=0;j<elements;j++){
    if(IsNonZero(pr[j]) || (cmplx && IsNonZero(pi[j]))) {
        nnz++;
    }
    }

    /* 获取输入参数的维数，为返回参数分配内存 */
    number_of_dims=mxGetNumberOfDimensions(prhs[0]);
    plhs[0]=mxCreateDoubleMatrix(nnz,number_of_dims,mxREAL);
```

```
        pind=mxGetPr(plhs[0]);

        /* 获取输入参数的维数 */
        dim_array=mxGetDimensions(prhs[0]);

        /* 遍历整个数组，若发现是非零值，则添加到输出数组中
        * 并记录其在MATLAB中的索引，因为MATLAB索引值从1开始，而C语言从0开始
        * 所以得到的索引值需加1
        */
        for(j=0;j<elements;j++) {
          if(IsNonZero(pr[j]) || (cmplx && IsNonZero(pi[j]))) {
                mwSize temp=j;
                mwSize k;
                for (k=0;k<number_of_dims;k++){
                  pind[nnz*k+count]=(double)((temp % (dim_array[k])) +1);
                  temp/=dim_array[k];
                  }
              count++;
          }
        }
    }
```

用MEX命令编译此文件后，在MATLAB控制窗口中输入以下矩阵：

```
    matrix = [ 3 0 9 0; 0 8 2 4; 0 9 2 4; 3 0 9 3; 9 9 2 0]
    matrix =
         3     0     9     0
         0     8     2     4
         0     9     2     4
         3     0     9     3
         9     9     2     0
```

调用findnz MEX文件，得到以下结果：

```
    nz = findnz(matrix)
    nz =
         1     1
         4     1
         5     1
         2     2
         3     2
         5     2
         1     3
         2     3
         3     3
         4     3
         5     3
         2     4
         3     4
         4     4
```

此MEX文件仅仅返回了非零值的索引，以列优先顺序排列。

（7）多个输入/输出参数

C语言中一个函数只可以返回一个值，而MATLAB不同，可以返回多于一个值。入口

子程序中的参数 plhs[]和 prhs[]都是向量，其元素分别维指向调用入口子程序时各个输出和输入参数的指针，比如 plhs[0]是指向第一个输出参数的指针，plhs[1]为指向第二个输出参数的指针，以此类推。类似的，prhs[0] 是指向第一个输入参数的指针，prhs[1]为是指向第二个输入参数的指针等。合理利用 plhs[]和 prhs[]两个指针数组就可以处理任意数目的输入/输出参数的情况。

例 10.10 编写 MEX 文件，实现一个标量和矩阵的乘法，并返回结构矩阵的功能。

源文件为 xtimesy.c，其内容如下：

xtimesy.c

```
#include "mex.h"

void xtimesy(double x, double *y, double *z, mwSize m, mwSize n)
{
  mwSize i,j,count=0;

  for (i=0; i<n; i++) {
    for (j=0; j<m; j++) {
      *(z+count) = x * *(y+count);
      count++;
    }
  }
}

void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[])
{
  double *y,*z;
  double  x;
  mwSize mrows,ncols;

  /*  检查输入参数的个数 */
  if(nrhs!=2)
    mexErrMsgTxt("Two inputs required."); // mexErrMsgTxt 语句会退出 MEX 文件的执行
  if(nlhs!=1)
    mexErrMsgTxt("One output required.");

  /* 第一个输入参数必须是标量 */
  if( !mxIsDouble(prhs[0]) || mxIsComplex(prhs[0]) ||
      mxGetN(prhs[0])*mxGetM(prhs[0])!=1 ) {
    mexErrMsgTxt("Input x must be a scalar.");
  }

  /*  获取标量参数 x */
  x = mxGetScalar(prhs[0]);

  /*  创建指针指向输入参数矩阵 y */
  y = mxGetPr(prhs[1]);
```

```
    /*  获取 y 的维数 */
    mrows = mxGetM(prhs[1]);
    ncols = mxGetN(prhs[1]);

    /*  输出参数指针指向输出矩阵 */
    plhs[0] = mxCreateDoubleMatrix(mrows,ncols, mxREAL);

    /*  创建 C 语言的指针指向输出矩阵的副本 */
    z = mxGetPr(plhs[0]);

    xtimesy(x,y,z,mrows,ncols);
}
```

注意第一个输入参数必须是标量，第二个是标量或矩阵，因为 MATLAB 中，标量也看做是 1×1 的矩阵。并且调用此 MEX 文件时，必须有输出参数，其内容如下：

```
x = 7;
y = 9;
z = xtimesy(x,y)
```

得到结果：

```
z =
    63
```

用一个标量和一个矩阵作为输入：

```
x = 3;
y = magic(5);
z = xtimesy(x,y)
```

得到结果为：

```
z =
    51    72     3    24    45
    69    15    21    42    48
    12    18    39    60    66
    30    36    57    63     9
    33    54    75     6    27
```

3）调用 MATLAB 函数

在 C 语言的 MEX 文件中，调用 MATLAB 的函数，比如运算符、M 文件以及其他 MEX 文件。MATLAB 的 API 函数 mexCallMATLAB 可以完成这一功能。

例 10.11 编写 MEX 文件，演示如何在 C 语言的 MEX 文件中调用 MATLAB 函数。

源文件为 sincall.c，其中调用 MATLAB 内嵌函数获取正弦波数据和正弦波绘图的两个语言如下：

```
mexCallMATLAB(1, lhs, 1, rhs, "sin");
mexCallMATLAB(0, NULL, 1, lhs, "plot");
```

此文件和上述的 timestwo.c 在同一目录下，调用编译后 MEX 文件：

```
sincall
```

得到如图 10-3 所示的图像。

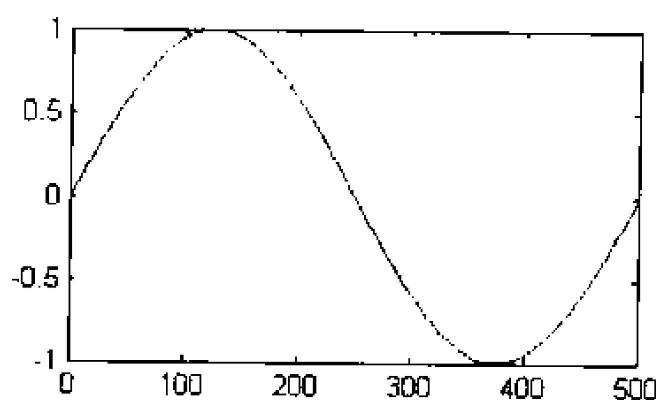


图 10-3 MEX 文件 sincall 的结果图

此外，MEX 文件无法提供帮助信息，所以若想为 MEX 文件编辑帮助信息，可以创建一个与此 MEX 文件同名的 M 文件，并放在同一目录下。在帮助文件的 M 文件中，由于功能是为同名的 MEX 文件提供帮助信息，所以不要有任何执行语句，而只是一些帮助信息，如解释此 MEX 的主要功能、对输入/输出参数的说明以及对算法的说明等。完成此 M 文件后，在 MATLAB 的控制窗口中输入 help 命令就可以查找对应的 MEX 文件的帮助信息。因为在 MATLAB 的解释器中，help 命令仅仅对命令中同名的 M 文件进行查找，而忽略对 MEX 的查找。

10.2.4 Visual C++中建立 MEX 文件及调试

在 MATLAB 环境中可以方便的编辑 MEX 文件，而编译 MEX 文件只需要一条 mex 命令，所以一般不需要 Visual C++的集成环境来建立 MEX 文件。但是对于大型的 MEX 程序，可能会比较复杂，容易出错，这时会考虑利用 Visual C++强大的调试功能。特别是 MEX 程序中与很多与操作系统有关的代码，如出现 Windows 对话框和访问、控制系统硬件等，在 Visual C++中创建 MEX 文件及调试尤为必要。本节利用一个十分简单的例子来说明 Visual C++中建立及调试的步骤。

1．Visual C++中 MEX 文件的建立

例 10.12 用 Visual C++创建 MEX 文件，功能为在 MATLAB 控制窗口输出一个字符串。

解题步骤如下：

（1）建立一个 DLL 工程

由于 MEX 程序实际上是一个动态链接库（DLL），其输出函数是 MEX 文件的入口函数 mexFunction。在 Visual C++中创建一个 MFC DLL 工程，命名为 MexDemo，下一步设置为默认设置，第一步如图 10-4 所示。

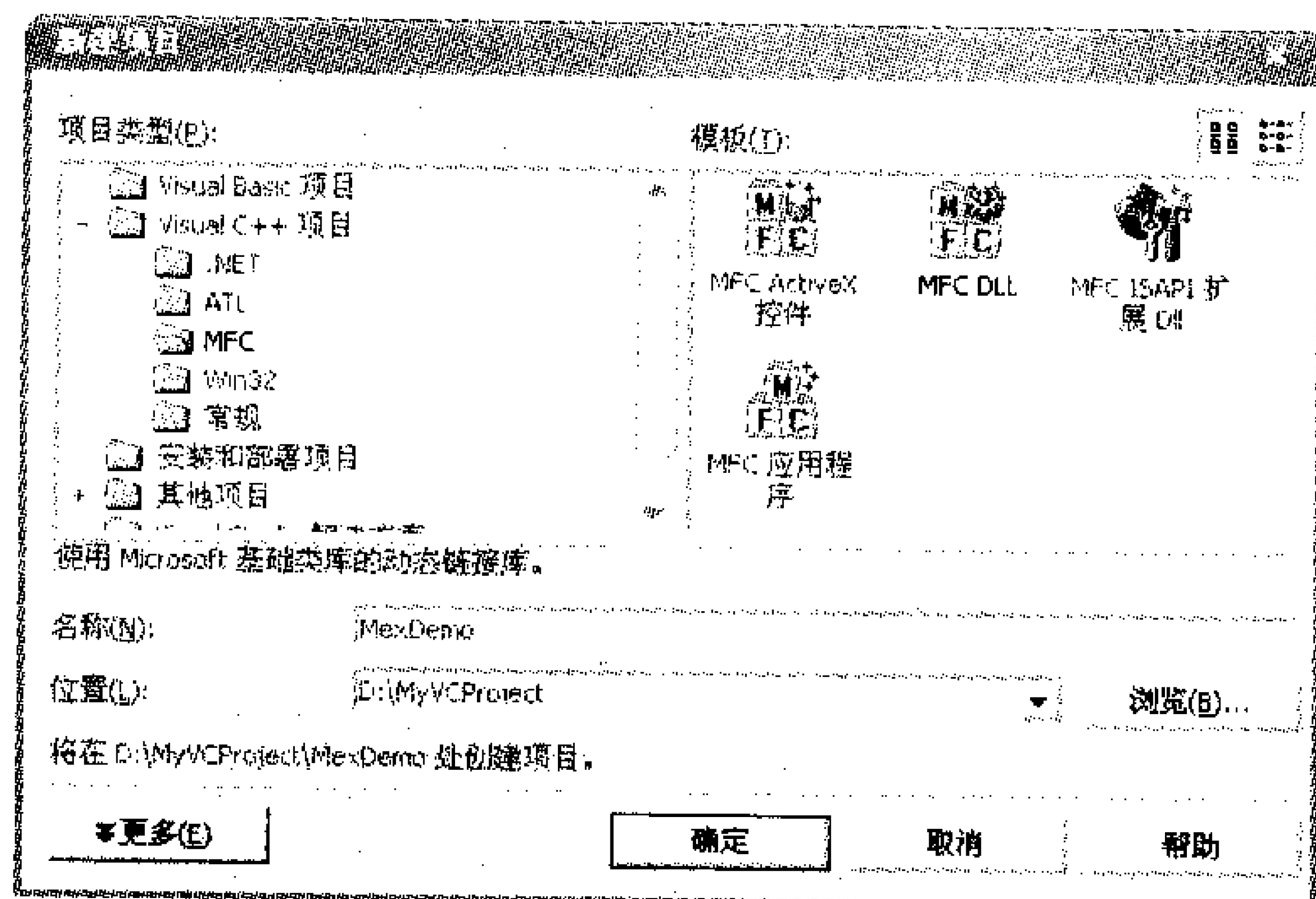


图 10-4 创建 MFC DLL 工程

（2）设置输出函数

成功 MFC DLL 工程后，会生成一个 MexDemo.def 文件，需要在其中定制 DLL 文件的输出函数，打开此文件，在 EXPORTS 项中添加 mexFunction 一行，设置后，MexDemo.def 文件内容如下：

```
; MexDemo.def : 声明 DLL 的模块参数。

LIBRARY      "MexDemo"

EXPORTS
    ; 此处可以是显式导出
    mexFunction
```

（3）添加应用程序代码

工程建立后，仅仅是一个空工程，没有实际功能，需要添加用户实现功能的代码。在工程中新建一个 C++源文件，命名为 main.cpp，并将此文件添加到此工程的源文件中。其中 main.cpp 没有计算子程序，仅有入口子程序，而功能是输出一行 “Hello MATLAB World!”，文件内容如下：

main.cpp

```
#include "stdafx.h"
#include "mex.h"

void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray *prhs[])
{
    mexPrintf("Hello MATLAB World!");
}
```

（4）编译器设置

由于此 C++文件有 mexPrintf 函数，并可能含有其他 MATLAB 库函数，所以要在 Visual C++中成功编译此类 MEX 文件，还需对 Visual C++的编译器作一些必要设置。下面以 Visual C++.NET 2005 为例做如下设置。

①添加 include 路径。通过执行【项目】→【属性】命令，打开工程属性设置对话框。在“配置属性”→“C/C++”→“常规”属性页，在“附加包含目录”选项中添加 MATLAB 外部接口的 include 目录，本机为“D:\MATLABR2007a\extern\include”。详细情况如图 10-5 所示。

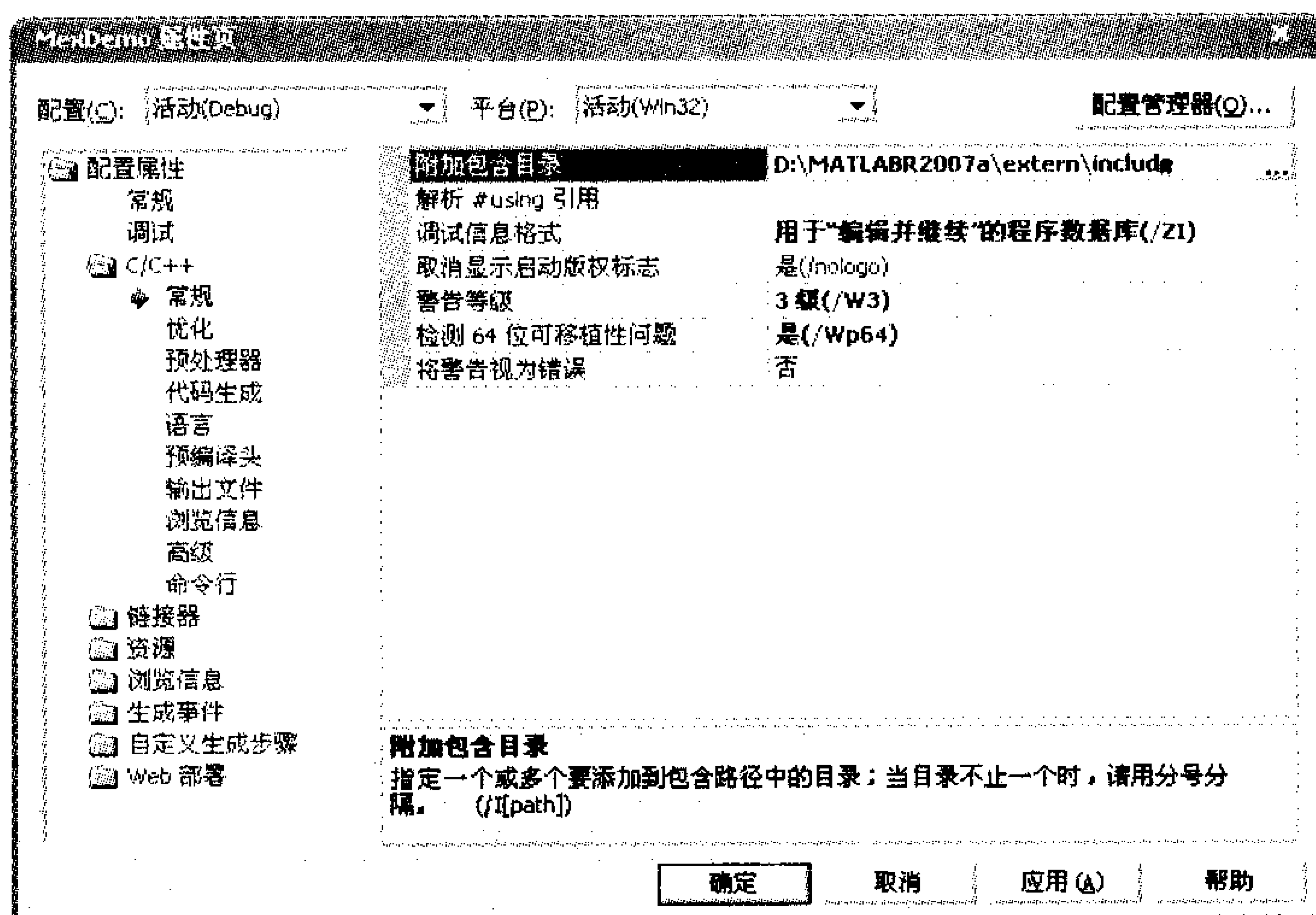


图 10-5　添加头文件目录

②添加 lib 路径。在打开

的 meshdemo 属性页中，展开“配置属性”→“链接器”→“常规”页，在“附加库目录”行添加 MATLAB 外部接口链接库的目录，本机为“D:\MATLABR2007a\extern\lib\win32\microsoft”。详细情况如图 10-6 所示。

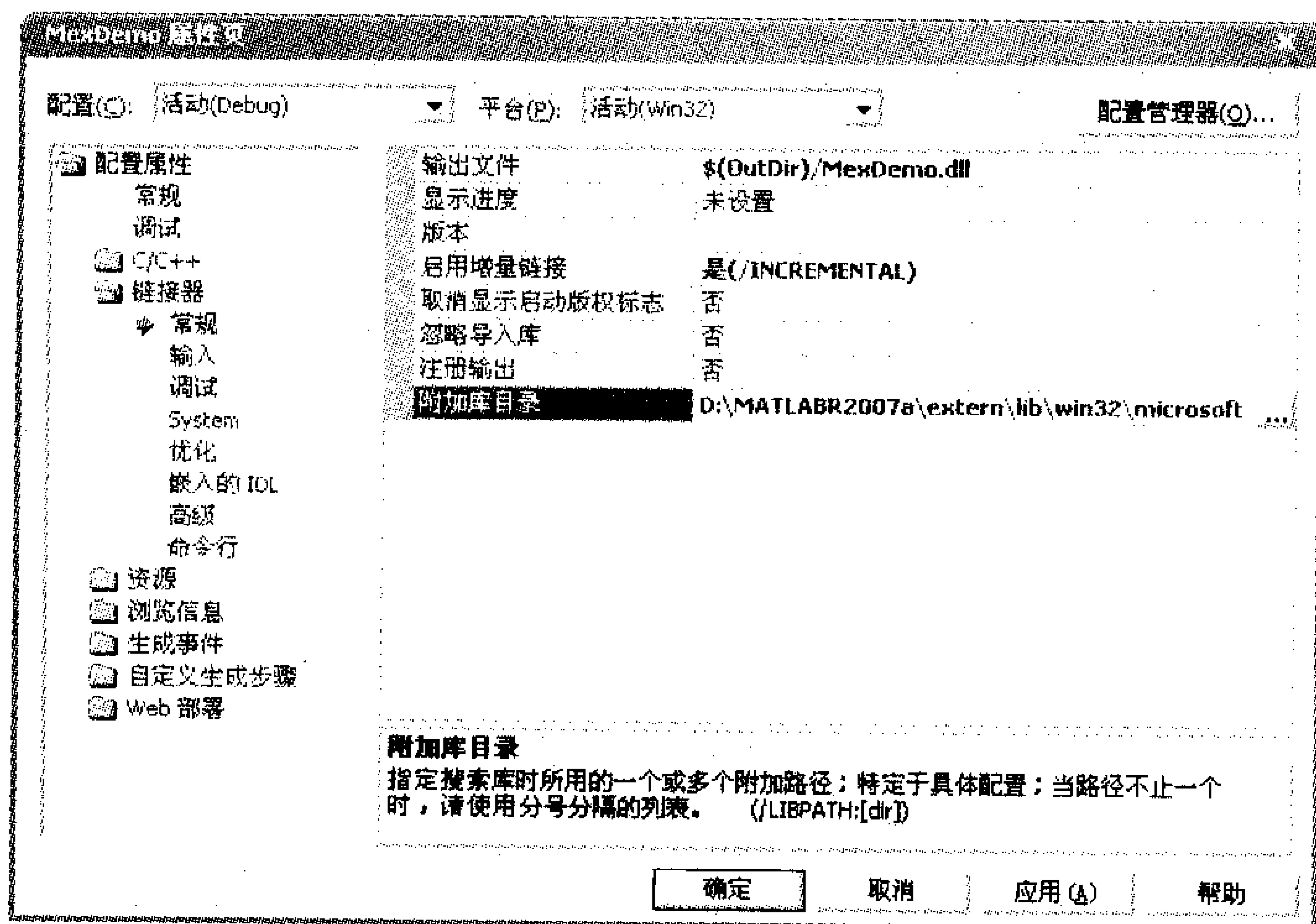


图 10-6 设置库文件包含目录

③添加必需的链接库。在打开的 meshdemo 属性页中，展开“配置属性”→“链接器”→“输入”页中的附加依赖项 libmx.lib、libeng.lib、libmat.lib 以及 libmex.lib，如图 10-7 所示。

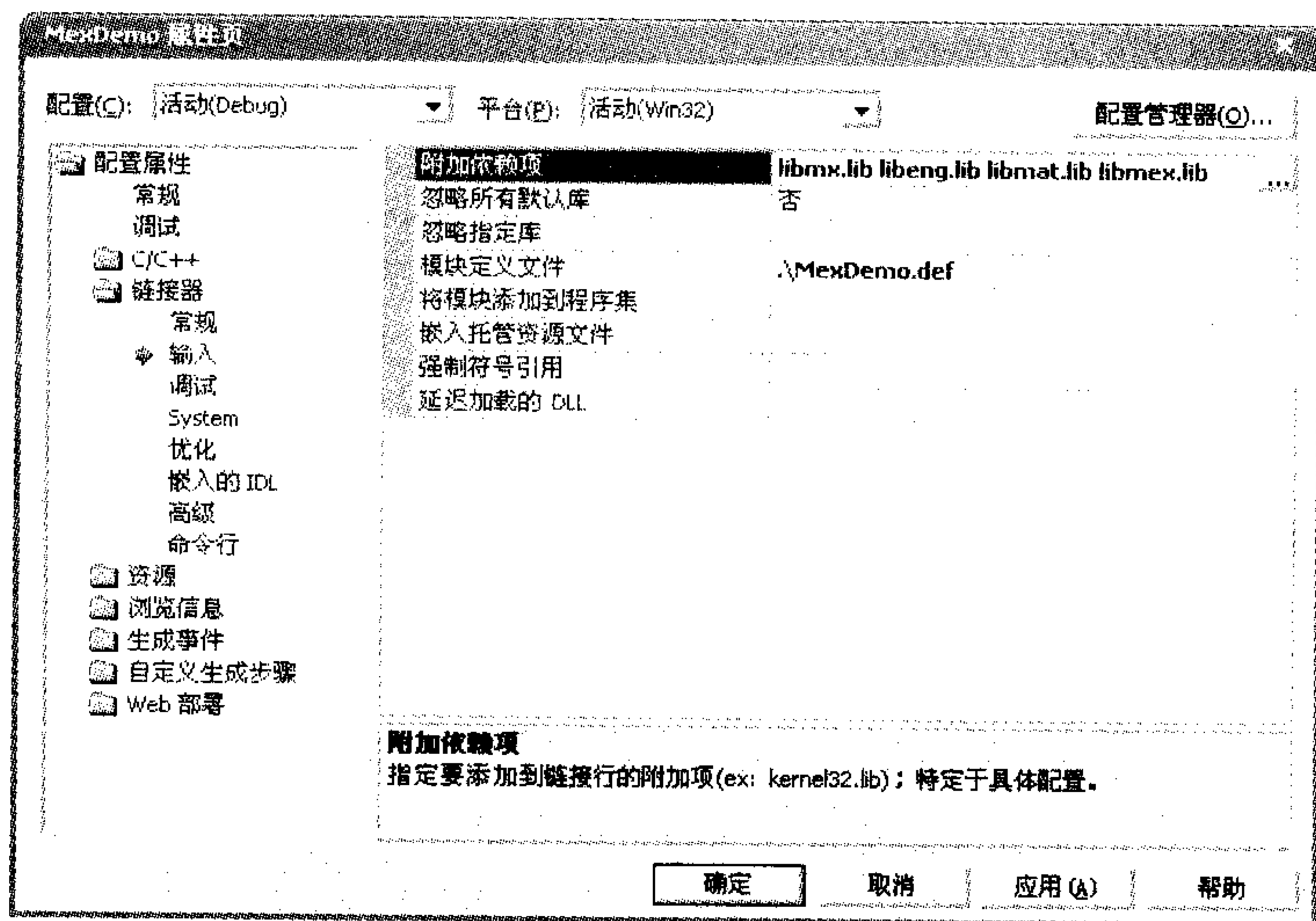


图 10-7 添加附加库文件

（5）编译

设置好 Vsual C++的编译器，就可以编译此工程了，编译成功后，在 debug 目录下生成 MexDemo.dll 链接库文件。

（6）运行

在 MATLAB 环境中，把此 debug 目录设为当前目录，然后在 MATLAB 控制窗口中输入 MexDemo，在 MATLAB 控制窗口得到结果：

```
Hello MATLAB World!>>
```

其中后面的>>表示等待用户输入。

2．Visual C++中 MEX 文件的调试

MEX 文件是 DLL 文件，所以调试比一般文件复杂，有时需要全面跟踪，所以此时可以利用 Visual C++的调试功能。

由于 MEX 文件不是 exe 文件，在调试时需要指定执行的命令。打开 MEXDemo 属性页，配置选择“活动(Debug)”，然后在“调试”选项卡中的“命令”行，找到本机中 MATLAB.exe 的目录。设置完毕后，如图 10-8 所示。

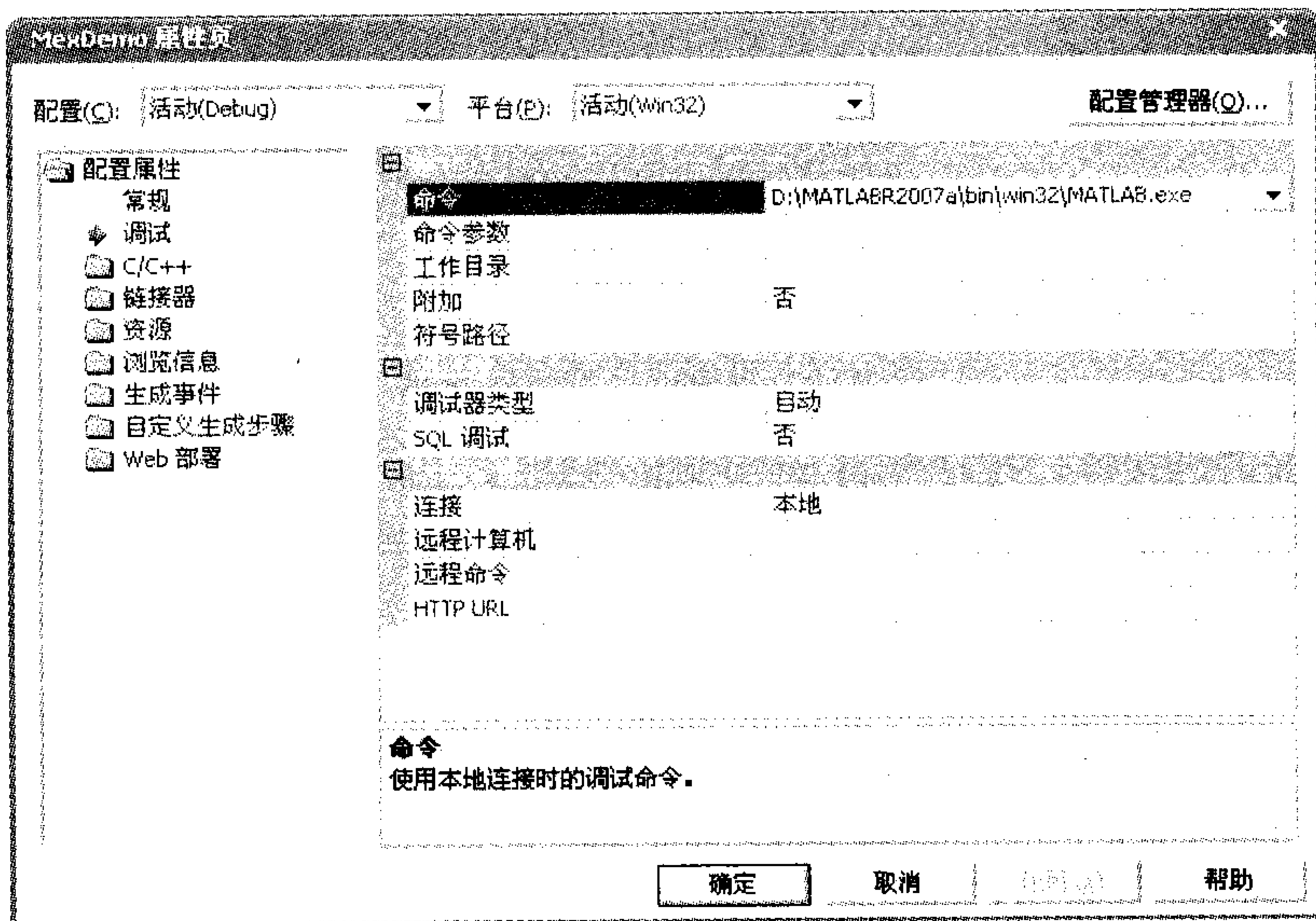


图 10-8　调试命令设置

正确设置后，可以采用在文件设置断点等调试方法。调试开始时，会出现如图 10-9 所示的警告信息，直接单击【确定】按钮即可。

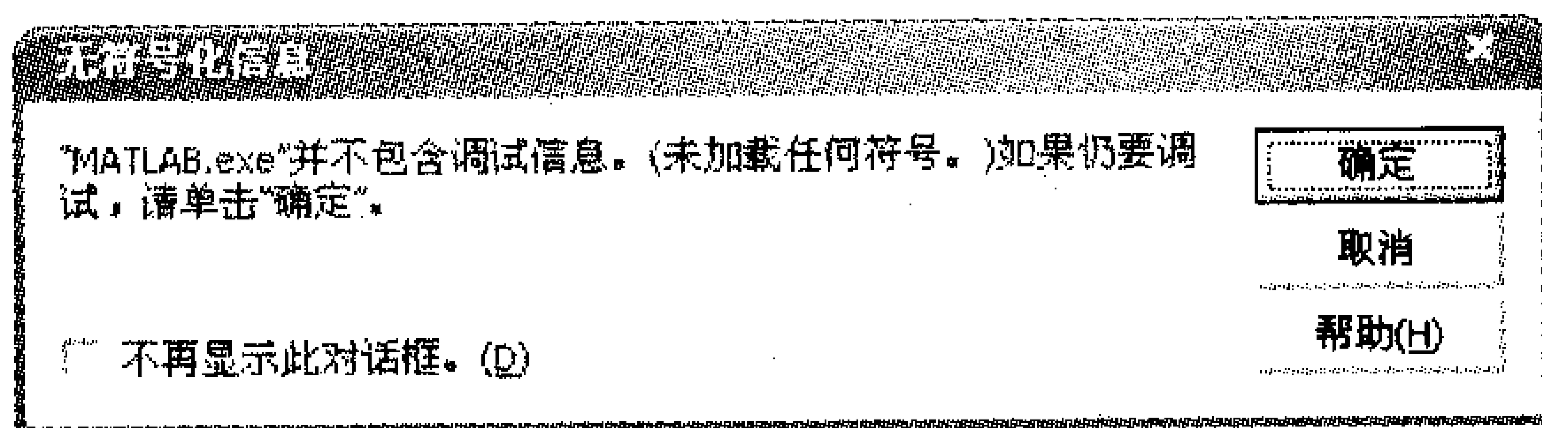


图 10-9　警告信息

调试开始后，会单独打开一个 MATLAB 进程，其默认当前路径是 MATLAB.exe 的路径，所以还要将调试的文件的目录设为当前目录，然后在控制窗口中输入要执行的文件，即可对其进行调试了。单击【停止调试】按钮，系统会自动关闭刚启动的 MATLAB 进程，并退出调试模式。

Visual C++中建立 MEX 文件的难点是编译器等相关系统设置，所以选择一个极为简单的例子，突出介绍 Visual C++中建立和调试 MEX 文件的步骤和方法，在掌握了步骤和方法后，读者不难处理较为复杂的 MEX 文件。



MEX 文件不是对所有的场合都适用，由于 MATLAB 本身是一个高效的系统，所以一般在 MATLAB 环境编程时尽量以 MATLAB 为主，只是对那些耗时或 MATLAB 功能受限而适用 MEX 文件可以较为显著提高效率的那部分采用 MEX 编程。

10.3 MATLAB MAT 文件

MAT 文件实现了 MATLAB 的程序和文件与其他编程环境的数据交换。MAT 文件是 MATLAB 默认的数据存储的文件格式，其数据以字节流的方式顺序写入，以二进制格式存储，这种格式为不同平台之间或 MATLAB 与不同应用程序之间共享数据提供了便利。

本节主要介绍了 MAT 文件的数据输入/输出、MAT 文件格式，最后以实例介绍了 MAT 文件编程。

10.3.1 数据输入/输出

MATLAB 中是以后缀为.mat 的文件，即 MAT 文件来存储工作空间的变量的，MAT 文件同时提供了一个数据交换机制，使得 MATLAB 的数据可在不同平台之间方便地交换数据。下面先介绍 MATLAB 常用的数据输入/输出方法。

1．MATLAB 数据输入

数据输入/输出有很多方法，这些方法的选择取决于数据的大小，数据是否为机器可读取的格式以及具体的数据文件格式。一般来说，数据输入有以下几种方法。

（1）在 MATLAB 的控制窗口中直接输入。

这种方法适用于比较少的数据，一般少于 15 个元素。当数据元素变多时，这种方法变得笨拙，因为在控制窗口输入烦琐和易出错而且出错后不便于编辑。

（2）在 M 文件中创建数据。

利用文本编辑器创建数据的列表，这种方法适用于数据还没有输入到计算机中时。用 M 文件便于编译和修改。

（3）Load 命令从 ASCII 平坦文件（Flat Files）中导入。

可用文本编辑器编辑 ASCII 平坦文件，然后用 load 命令导入数据。

（4）用 MATLAB 的 I/O 函数。

用 fopen、fread 以及其他低层 MATLAB I/O 函数读取数据。这种方法可以读取其他文件格式中的数据。

（5）编写 MEX 文件。

编辑成 MEX 文件子程序完成读写数据功能，以供 MATLAB 外的程序调用。

（6）编写 C 程序或 Fortran 程序，将数据转换成 MAT 文件格式，然后调用 load 命令将数据导入到 MATLAB。

2．MATLAB 数据输出

从 MATLAB 导出数据有下列几种方法。

（1）创建日志文件（Diary File）。

使用命令：

```
diary filename
```

可以创建一个日志文件，若不指定文件名，则默认文件名为 diary，随后在控制窗口输入的任何命令都会记录在此文件中。使用文本编辑器可以编辑修改此文件。最后用命令 diary off 来终止记录。

（2）使用 save 命令。

使用-ascii 选项的 save 命令可以将数据保存为 ASCII 格式的文件。内容如下：

```
A = rand(4,3);
save temp.dat A –ascii
```

创建名为 temp.dat 的 ASCII 文件。内容如下：

```
1.3889088e-001   2.7218792e-001   4.4509643e-001
2.0276522e-001   1.9881427e-001   9.3181458e-001
1.9872174e-001   1.5273927e-002   4.6599434e-001
6.0379248e-001   7.4678568e-001   4.1864947e-001
```

（3）使用 MATLAB 的 I/O 函数。

使用 fopen、fwrite 以及其他低层文件 I/O 函数可以操作常见的数据文件。

（4）创建 MEX 文件。

在 MATLAB 外的程序中，创建 MEX 文件完成读数据的子功能供其他程序调用。

（5）C 程序或 Fortran 程序转换 MAT 文件。

编辑 C 程序或 Fortran 程序来转换 MAT 文件为特定格式的文件。

3．不同平台间数据交换

用户可在不同系统上运行 MATLAB，也可向不同系统的计算机发送 MATLAB 应用程序，其中应用程序包括脚本/函数 M 文件和包含数据的 MAT 文件。

上述两种 MATLAB 应用程序都可以直接在不同的机器间移植：M 文件是与平台无关的，而 MAT 文件的文件头包含了系统信息。MATLAB 导入 MAT 文件时，先检测其包含的系统信息，若发现此 MAT 文件的系统与本系统不同，则进行一些必要的转换。

在不同机器系统结构中使用 MATLAB 时，需要转换此机器上二进制和 ASCII 格式数据的工具，此类工具包括 FTP、NFS 以及 Kermit。使用这些工具时，传送 MAT 文件要以二进制格式，M 文件以 ASCII 格式，若格式设置不正确可能为损坏数据。

10.3.2 MAT 文件格式

MAT 文件是 MATLAB 专门的数据存储方式，是一种二进制文件。MAT 数据格式是 MATLAB 数据存储的标准格式，有其独特的存储格式。MAT 文件由 128 字节的文件头和其后的具体的数据单元组成，而每个数据单元头都有一个 8 字节的标识，这个标识中描述了此数据单元的数据类型和数据长度（大小）。

具体的 MAT 文件存储格式如图 10-10 所示。

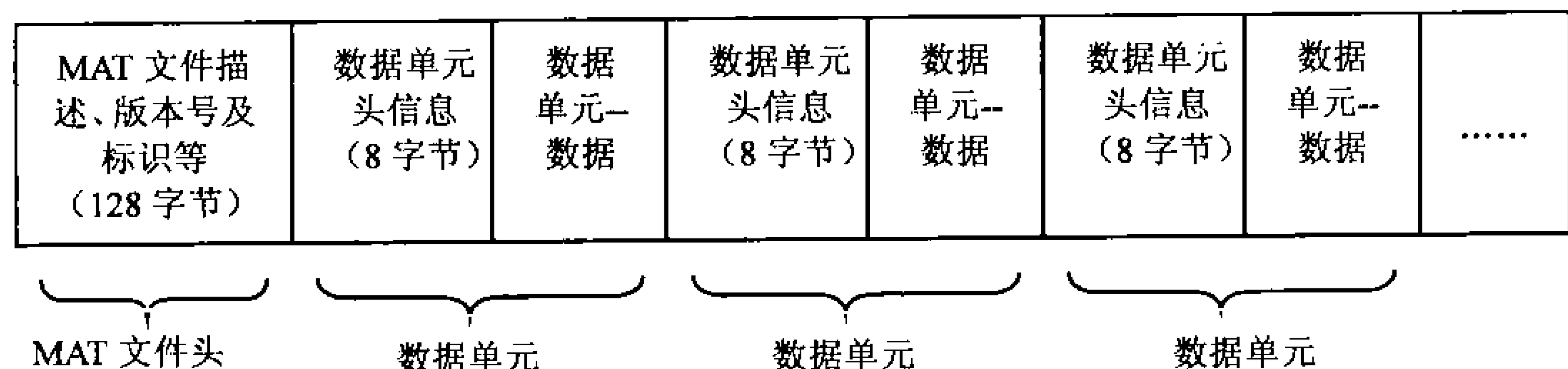


图 10-10　MAT 文件数据格式

其中 MAT 文件头信息为文本信息，可以用命令来查询。在 Windows 系统中，可以在 MATLAB 控制窗口中输入 type 命令查询，内容如下：

```
type matlab.mat
```

得到下列信息：

```
MATLAB 5.0 MAT-file, Platform: PCWIN, Created on: Fri Jul 06 11:07:03 2007
```

表示 MAT 文件为 5.0 版本，平台为 PCWIN，创建时间为 Fri Jul 06 11:07:03 2007。

10.3.3　MAT 文件编程

一般情况下，用户不需要了解 MAT 文件的格式，MATLAB 提供了一些在 MATLAB 外的程序中也可以操作 MAT 文件的函数库，利用这些函数可以在 C 程序或 Fortran 程序中对 MAT 文件进行读写等操作。这些 API 函数都以 mat 为前缀，在 mat.h 中定义。

建议用 MATLAB 提供的这些 API 函数库，而不是用自己编写的函数，因为 MAT 文件升级后，自己编写的读写 MAT 文件程序必须重写。如表 10-2 所示介绍了 mat.h 定义的操作 MAT 文件的 API 函数。

表 10-2　C 语言操作 MAT 文件的函数

函数	用途
matOpen	打开 MAT 文件
matClose	关闭 MAT 文件
matGetDir	获得 MAT 文件的变量列表
matGetFp	获得 MAT 文件的 C 指针
matGetVariable	从 MAT 文件中读取变量
matPutVariable	将变量写入 MAT 文件
matGetNextVariable	读取 MAT 下一个变量
matDeleteVariable	删除 MAT 文件的一个变量
matPutVariableAsGlobal	将变量写入 MAT 文件，当“load”导入此 MAT 文件时，此变量为 global 型
matGetVariableInfo	从 MAT 文件载入变量头（不载入此变量）
matGetNextVariableInfo	载入下一个变量的头信息

在 Visual C++中创建工程后，要正确设置编译器才能完成编译和生成可执行文件。如表 10-3 所示列出了 Windows 平台和 UNXT 平台下需要的头文件目录和库文件目录。

表 10-3 编译器设置

Windows	Include files	matlabroot\extern\include
	Libraries	matlabroot\bin\win32（32 位机） matlabroot\bin\win64（64 位机）
UNIX	Include files	matlabroot/extern/include
	Libraries	matlabroot/bin/$arch

其中 libraries 选项中，matlabroot\bin\win32 下还有 4 个子目录：microsoft、borland、lcc 和 watcom，若是使用 microsoft 的 Visual C++等产品，则选择目录 microsoft。下面以一个实例在介绍如何在 Visual C++中建立创建工程并调试。

例 10.13 用 MATLAB 的 API 函数创建一个能导入到 MATLAB 环境中的 MAT 文件。

（1）在 Visual C++中创建一个空的 Win32 控制台工程。

将 matlabroot\extern\examples\eng_mat \matcreat.c 文件复制到工程的目录下，并设置为工程的源文件，其文件内容为（略有修改）：

matcreat.c

```
#include <stdio.h>
#include <string.h>  /* 要使用到字符串操作函数 strcmp()            */
#include <stdlib.h>  /* 要用到 EXIT_FAILURE, EXIT_SUCCESS        */
#include "mat.h"      /* 操作 MAT 文件 API 函数的头文件，必须添加 */

#define BUFSIZE 256

int main() {
  /* 创建指向 MAT 文件类型的指针 */
  MATFile *pmat;
/* 创建必要的变量 */
  mxArray *pa1, *pa2, *pa3;
  double data[9] = { 1.0, 4.0, 7.0, 2.0, 5.0, 8.0, 3.0, 6.0, 9.0 };
  const char *file = "mattest.mat";
  char str[BUFSIZE];
  int status;

  printf("Creating file %s...\n\n", file);
  /* 以写方式打开 MAT 文件，若文件不存在，则创建一个并打开 */
  pmat = matOpen(file, "w");
  /* 检查返回值，若打开失败，返回 */
  if (pmat == NULL) {
    printf("Error creating file %s\n", file);
    printf("(Do you have write permission in this directory?)\n");
    return(EXIT_FAILURE);
  }

  /* 创建 3×3 的矩阵 */
  pa1 = mxCreateDoubleMatrix(3,3,mxREAL);
  if (pa1 == NULL) {
```

```
        printf("%s : Out of memory on line %d\n", __FILE__, __LINE__);
        printf("Unable to create mxArray.\n");
        return(EXIT_FAILURE);
    }

    pa2 = mxCreateDoubleMatrix(3,3,mxREAL);
    if (pa2 == NULL) {
        printf("%s : Out of memory on line %d\n", __FILE__, __LINE__);
        printf("Unable to create mxArray.\n");
        return(EXIT_FAILURE);
    }
    /* 用 data 的数据初始化 pa3 */
    memcpy((void *)(mxGetPr(pa2)), (void *)data, sizeof(data));

    /* 创建字符串变量 */
    pa3 = mxCreateString("MATLAB: the language of technical computing");
    if (pa3 == NULL) {
        printf("%s :  Out of memory on line %d\n", __FILE__, __LINE__);
        printf("Unable to create string mxArray.\n");
        return(EXIT_FAILURE);
    }

    /* 将 pa1 指向的数据写入 MAT 文件中 */
    status = matPutVariable(pmat, "LocalDouble", pa1);
    if (status != 0) {
        printf("%s :  Error using matPutVariable on line %d\n", __FILE__, __LINE__);
        return(EXIT_FAILURE);
    }

  /* 将 pa2 指向的数据写入 MAT 文件中 */
    status = matPutVariableAsGlobal(pmat, "GlobalDouble", pa2);
    if (status != 0) {
        printf("Error using matPutVariableAsGlobal\n");
        return(EXIT_FAILURE);
    }

    /* 将 pa3 指向的数据写入 MAT 文件中 */
    status = matPutVariable(pmat, "LocalString", pa3);
    if (status != 0) {
        printf("%s :  Error using matPutVariable on line %d\n", __FILE__, __LINE__);
        return(EXIT_FAILURE);
    }

    /* 下面代码演示了 matPutVariable 会覆盖 MAT 文件中已有的数据  */
    memcpy((void *)(mxGetPr(pa1)), (void *)data, sizeof(data));
    status = matPutVariable(pmat, "LocalDouble", pa1);
    if (status != 0) {
        printf("%s :  Error using matPutVariable on line %d\n", __FILE__, __LINE__);
        return(EXIT_FAILURE);
    }
```

```
  /* 清理数据 */
  mxDestroyArray(pa1);
  mxDestroyArray(pa2);
  mxDestroyArray(pa3);

  /* 关闭 MAT 文件 */
  if (matClose(pmat) != 0) {
    printf("Error closing file %s\n",file);
    return(EXIT_FAILURE);
  }

  /* 以只读方式重新打开 MAT 文件，用 matGetVariable 函数验证其内容 */
  pmat = matOpen(file, "r");
  if (pmat == NULL) {
    printf("Error reopening file %s\n", file);
    return(EXIT_FAILURE);
  }

  /* 读取刚刚写入的变量 pa1、pa2、pa3，检查是否已成功写入 */
  pa1 = matGetVariable(pmat, "LocalDouble");
  if (pa1 == NULL) {
    printf("Error reading existing matrix LocalDouble\n");
    return(EXIT_FAILURE);
  }
  if (mxGetNumberOfDimensions(pa1) != 2) {
    printf("Error saving matrix: result does not have two dimensions\n");
    return(EXIT_FAILURE);
  }

  pa2 = matGetVariable(pmat, "GlobalDouble");
  if (pa2 == NULL) {
    printf("Error reading existing matrix GlobalDouble\n");
    return(EXIT_FAILURE);
  }
  if (!(mxIsFromGlobalWS(pa2))) {
    printf("Error saving global matrix: result is not global\n");
    return(EXIT_FAILURE);
  }

  pa3 = matGetVariable(pmat, "LocalString");
  if (pa3 == NULL) {
    printf("Error reading existing matrix LocalString\n");
    return(EXIT_FAILURE);
  }

  status = mxGetString(pa3, str, sizeof(str));
  if(status != 0) {
      printf("Not enough space. String is truncated.");
      return(EXIT_FAILURE);
```

```
    }
    if (strcmp(str, "MATLAB: the language of technical computing")) {
      printf("Error saving string: result has incorrect contents\n");
      return(EXIT_FAILURE);
    }

    /* 退出前清理临时变量 */
    mxDestroyArray(pa1);
    mxDestroyArray(pa2);
    mxDestroyArray(pa3);

    /* 关闭 MAT 文件 */
    if (matClose(pmat) != 0) {
      printf("Error closing file %s\n",file);
      return(EXIT_FAILURE);
    }
    printf("Done\n");
    return(EXIT_SUCCESS);
}
```

此源文件虽然比较长，但是结构清晰，内容易懂。从中容易学会 matClose、matGetVariable、matOpen、matPutVariable 和 matPutVariableAsGlobal 等 API 函数的使用方法。

（2）设置编译器，按照 10.2.4 节中的编译器设置方法设置好包含目录和库目录，并添加库文件 libmat.lib 和 libmx.lib。设置好以后就可以运行查看结果了。

（3）运行，验证结果。运行此程序，会显示如下内容：

```
Creating file mattest.mat...
Done
```

表明成功生成 matest.mat 文件，而且读写过程中没有出错。此文件可以导入到 MATLAB 基本工作空间中。为验证此 MAT 文件的内容，在 MATLAB 的控制窗口中输入：

```
whos -file mattest.mat
```

得到下面信息：

```
  Name            Size      Bytes    Class     Attributes
  GlobalDouble    3x3       72       double    global
  LocalDouble     3x3       72       double
  LocalString     1x43      86       char
```

以上结果表明 MAT 文件成功创建，而且数据写入无误。

例 10.14 利用 MATLAB 的 API 函数读取已有 MAT 文件信息及数据。

以 MATLAB 自带的例子文件 matdgns.c 为源文件和上例一样建立 Win32 控制台工程，编译器设置也相同，此文件和 creatmat.c 在同一个目录下，其内容如下：

matdgns.c

```
#include <stdio.h>
#include <stdlib.h>
#include "mat.h"

int diagnose(const char *file) {
```

```
MATFile *pmat;
const char **dir;
const char *name;
int       ndir;
int       i;
mxArray *pa;

printf("Reading file %s...\n\n", file);

/* 以只读方式打开 MAT 文件 */
pmat = matOpen(file, "r");
if (pmat == NULL) {
  printf("Error opening file %s\n", file);
  return(1);
}

/* 获取 MAT 文件的路径 */
dir = (const char **)matGetDir(pmat, &ndir);
if (dir == NULL) {
  printf("Error reading directory of file %s\n", file);
  return(1);
} else {
  printf("Directory of %s:\n", file);
  for (i=0; i < ndir; i++)
    printf("%s\n",dir[i]);
}
mxFree(dir);

/* 为正确使用 matGetNextXXX 函数，关闭 MAT 文件，然后重新以只读方式打开 */
if (matClose(pmat) != 0) {
  printf("Error closing file %s\n",file);
  return(1);
}
pmat = matOpen(file, "r");
if (pmat == NULL) {
  printf("Error reopening file %s\n", file);
  return(1);
}

/* 获取所有变量的信息 */
printf("\nExamining the header for each variable:\n");
for (i=0; i < ndir; i++) {
  pa = matGetNextVariableInfo(pmat, &name);
  if (pa == NULL) {
  printf("Error reading in file %s\n", file);
  return(1);
  }
  /* 诊断 pa 的头信息 */
  printf("According to its header, array %s has %d dimensions\n",
     name, mxGetNumberOfDimensions(pa));
```

```
        if (mxIsFromGlobalWS(pa))
            printf("    and was a global variable when saved\n");
        else
            printf("    and was a local variable when saved\n");
        mxDestroyArray(pa);
    }

    /* 关闭 MAT 文件，重新打开，读取实际数据 */
    if (matClose(pmat) != 0) {
        printf("Error closing file %s\n",file);
        return(1);
    }
    pmat = matOpen(file, "r");
    if (pmat == NULL) {
        printf("Error reopening file %s\n", file);
        return(1);
    }

    /* 依次读取每个数据 */
    printf("\nReading in the actual array contents:\n");
    for (i=0; i<ndir; i++) {
          pa = matGetNextVariable(pmat, &name);
          if (pa == NULL) {
          printf("Error reading in file %s\n", file);
          return(1);
          }

          /* 诊断 Diagnose 变量 pa */
          printf("According to its contents, array %s has %d dimensions\n",
                 name, mxGetNumberOfDimensions(pa));
          if (mxIsFromGlobalWS(pa))
        printf("    and was a global variable when saved\n");
          else
        printf("    and was a local variable when saved\n");
          mxDestroyArray(pa);
    }

    if (matClose(pmat) != 0) {
          printf("Error closing file %s\n",file);
          return(1);
    }
    printf("Done\n");
    return(0);
}

int main(int argc, char **argv)
{
    int result;

    if (argc > 1)
```

```
            result = diagnose(argv[1]);
        else{
            result = 0;
            printf("Usage: matdgns <matfile>");
            printf(" where <matfile> is the name of the MAT-file");
            printf(" to be diagnosed\n");
        }

        return (result==0)?EXIT_SUCCESS:EXIT_FAILURE;
    }
```

此文件同样结构清晰，内容易懂。编译链接生成可执行程序后，以下面的格式调用此程序读取名为 matfile 的 MAT 文件的信息：

```
matdgns  <matfile>
```

此程序也很好地演示了操作 MAT 文件 matGetDir、matGetNextVariable 以及 matGetNextVariableInfo 函数。

上面简单介绍了在 Visual C++中创建工程，利用 MATLAB 的 API 函数库创建和读取 MAT 文件。由于 MATLAB 的 API 函数屏蔽了 MAT 文件存储格式，所以在 Visual C++中创建工程操作 MAT 文件变得简单。通过 MAT 文件操作，可以在不同平台上共享 MATLAB 数据，进而可以利用 MATLAB 的数值计算和图像显示处理功能。

10.4 MATLAB 计算引擎

MATLAB 引擎库（Engine Library）是 MATLAB 提供的一组函数库和程序库，在其他的非 MATLAB 程序中，如 C 语言和 Fortran 语言程序中可以调用这些函数。这种模式下，MATLAB 作为计算引擎，与 MATLAB 进程进行通信的 C 程序或 Fortran 程序称为 MATLAB 引擎程序。MATLAB 引擎实现了其他程序与 MATLAB 进程的交互，完成了二者之间的数据交换和命令传送的任务。

本节讨论了 MATLAB 的计算引擎库、举例说明如何在 C 语言和 Fortran 语言中调用 MATLAB 引擎，最后实例介绍建立与调试 MATLAB 计算引擎程序。

10.4.1 MATLAB 计算引擎

MATLAB 引擎（Engine）是指一组 MATLAB 提供的接口函数，支持 C/C++和 Fortran 语言等，通过这些接口函数，用户可以在其他编程环境中实现对 MATLAB 的控制：可以完成启动/关闭一个 MATLAB 引擎、向 MATLAB 环境发送命令字符串、从 MATLAB 环境中读取数据以及向 MATLAB 环境中写入数据等功能。

与其他各种接口相比，引擎所提供的 MATLAB 功能支持是最全面的。通过引擎方式，应用程序会打开一个新的 MATLAB 进程，可以控制它完成任何计算和绘图操作。对所有的 MATLAB 数据结构都提供了完全支持。

实际上，其他程序通过引擎方式与 MATLAB 进程建立的对话，是将 MATLAB 以 ActiveX 控件方式启动的。在 MATLAB 初次安装时，会自动执行以下命令

MATLAB /regserver

以在系统的控件库中注册 MATLAB。当无法打开 MATLAB 引擎时，可以尝试重新注册，方法为在 DOS 命令提示符后执行上述命令。

利用 MATLAB 引擎，可以完成下面功能。

- 调用数学计算子程序。如反转一个数组、求矩阵的特征值以及傅里叶转换。这种模式下，MATLAB 就称为一个功能强大，可编程数学子程序库。
- 为特殊任务创建一个系统。如已经成功开发的雷达信号分析系统和气体色谱法。这些系统中前端的图形用户界面用 C 语言编写，后台的数据分析调用 MATLAB，从而大大地缩短了开发时间。

MATLAB 以引擎模式运行时，作为一个独立的后台进程，前台用户界面由用户程序完成。这种在后台运行的模式有诸多好处，如 UNIX 系统中，MATLAB 引擎可以在本机上运行或网络上其他的 UNIX 系统机器上运行，包括不同系统结构的机器。可以在本机上执行图形交互界面，而在网络上其他功能更强，速度更快的机器上完成实际的计算功能。另外，以引擎模式运行时，MATLAB 不必与程序完全链接，而是仅仅一些负责通信和计算库链接即可，从而节省了系统资源。

10.4.2　计算引擎库函数

下面介绍 MATLAB 提供的引擎 API 函数，由于 C 语言的流行与高效，最后详细介绍 C 语言引擎 API 的使用方法。

1．C 语言和 Fortran 语言引擎 API

MATLAB 提供了操作 MATLAB 引擎的函数库，包括 C 语言和 Fortran 语言的。这个函数库可以完成启动/关闭 MATLAB 进程以及与 MATLAB 交换数据等功能。这些函数都以 eng 为前缀。如表 10-4 和表 10-5 所示分别列出了 C 语言和 Fortran 语言中的引擎函数及其用途。

表 10-4　C 语言计算引擎控制函数

函数	用途
engOpen	启动 MATLAB 引擎
engClose	关闭 MATLAB 引擎
engGetVariable	从 MATLAB 引擎获取 MATLAB 数组
engPutVariable	向 MATLAB 引擎发送 MATLAB 数组
engEvalString	执行 MATLAB 命令
engOutputBuffer	创建缓冲以存储 MATLAB 文本输出
engOpenSingleUse	启动一个专用的 MATLAB 引擎
engGetVisible	获取 MATLAB 引擎是否可见
engSetVisible	设置 MATLAB 引擎是否可见

表 10-5　Fortran 语言计算引擎控制函数

函数	用途
engOpen	启动 MATLAB 引擎
engClose	关闭 MATLAB 引擎
engGetVariable	从 MATLAB 引擎获取 MATLAB 数组
engPutVariable	向 MATLAB 引擎发送 MATLAB 数组
engEvalString	执行 MATLAB 命令
engOutputBuffer	创建缓冲以存储 MATLAB 文本输出

2. C 语言引擎 API 详解

下面对 C 语言的引擎 API 详细介绍，Fortran 语言中使用方法类似。

在调用 MATLAB 引擎之前，首先应在相关文件中加入如下代码：

```
#include "enging.h"
```

该文件包含了引擎 API 函数的说明和所需数据结构的定义，加入这行代码才能在 C 程序中调用下面的引擎 API 函数。在 Visual C++中调用的引擎函数分别如下。

1）engOpen

函数功能：启动 MATLAB 引擎。

函数声明：

```
Engine *engOpen(const char *startcmd);
```

参数说明：startcmd 是用来启动 MATLAB 引擎的字符串参数，在 Windows 操作系统中只能为 NULL。

返回参数：函数返回值是一个 engine 类型的指针，engine 数据结构在 engine.h 中定义。

2）EngClose

函数功能：关闭 MATLAB 引擎。

函数声明：

```
int engClose(Engine *ep);
```

参数说明：参数 ep 代表要被关闭的引擎指针。

返回参数：函数返回值为 0 表示关闭成功，返回 1 表示发生错误。

例如，通常用来打开/关闭 MATLAB 引擎的代码如下：

```
/* 定义 MATLAB 引擎指针*/
Engine *ep;
/* 测试是否启动 MATLAB 引擎成功 */
if (!(ep=engOpen(NULL)))。
{
    MessageBox("Can't start MATLAB engine!" );
    exit(1);
}

/* 可以调用 MATLAB 的其他函数完成计算操作 */

/* 最后，关闭 MATLAB 引擎 */
engClose(ep);。
```

3）engEvalString

函数功能：向 MATLAB 发送命令字符串让其执行。

函数声明：

```
int engEvalString(Engine *ep, Const char *string);
```

参数说明：参数 ep 为函数 engOpen 返回的引擎类型指针，字符串 string 为要 MATLAB 执行的命令。

返回参数：函数返回值为 0 表示成功执行，返回 1 说明执行失败（如命令不能被 MATLAB 正确解释或 MATLAB 引擎已经关闭）。

4）engOutputBuffer

函数功能：获取 MATLAB 命令窗口的输出。

要在 Visual C++中获得函数 engEvalString 发送的命令字符串被 MATLAB 执行后在 MATLAB 窗口中的输出，可以调用此函数。

函数声明：

```
int engOutputBuffer(Engine *ep, char *p, int n);
```

参数说明：参数 ep 为 MATLAB 引擎指针，p 为用来保存输出结构的缓冲区，n 为最大保存的字符个数，通常就是缓冲区 p 的大小。该函数执行后，接下来的 engEvalString 函数所引起的命令行输出结果会在缓冲区 p 中保存。如果要停止保存，只需调用如下代码：

```
engOutputBuffer(ep, NULL, 0)
```

返回参数：函数返回值为 0 表示成功执行，返回 1 说明执行失败。

5）engGetVariable

函数功能：从 MATLAB 引擎工作空间中获取变量。

函数声明：

```
mxArray *engGetVariable(Engine *ep, const char *name);
```

参数说明：参数 ep 为打开的 MATLAB 引擎指针，name 为以字符串形式指定的数组名。

返回参数：函数返回值是指向 name 数组的指针，类型为 mxArray*类型。

6）engPutVariable

函数功能：向 MATLAB 引擎工作空间写入变量。

函数声明：

```
int engPutVariable(Engine *ep, const char *name, const mxArray *mp);
```

参数说明：参数 ep 为打开的 MATLAB 引擎指针，mp 为指向被写入变量的指针，name 为变量写入后在 MATLAB 引擎工作空间中的变量名。

返回参数：函数返回值为 0 表示写入变量成功，返回值为 1 表示发生错误。

7）engSetVisible

函数功能：设置调用引擎时显示/隐藏 MATLAB 主窗口。

默认情况下，以引擎方式调用 MATLAB 的时候，会打开 MATLAB 主窗口，可在其中随意操作。但有时也会干扰应用程序的运行，此函数可以设置是否显示该窗口。

函数声明：

```
int engSetVisible(Engine *ep, bool value);
```

参数说明：参数 ep 为打开的 MATLAB 引擎指针，value 为是否显示的标志，取值 true（或 1）表示显示 MATLAB 窗口，取值 false（或 0）表示隐藏 MATLAB 窗口。

返回参数：函数返回值为 0 表示设置成功，为 1 表示有错误发生。

8）engGetVisible

函数功能：获取当前 MATLAB 窗口的显示/隐藏情况。

函数声明：

```
int engGetVisible(Engine *ep, bool *value);
```

参数说明：参数 ep 为打开的 MATLAB 引擎指针，Value 为用来保存显示/隐藏情况的变量采用指针方式传递。

返回参数：函数返回值为 0 表示获取成功，为 1 表示有错误发生。

10.4.3 计算引擎编程

下面以 MATLAB 的两个例子，程序 engdemo.c 和 fengdemo.f 来分别说明 C 语言和 Fortran 语言中调用 MATLAB 计算引擎。

1. C 程序调用 MATLAB 引擎

例 10.15 演示标准独立的 C 程序中调用 MATLAB 引擎。

例子文件为 engdemo.c，在 matlabroot\extern\examples\eng_mat 目录下，内容如下：

engdemo.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "engine.h"
#define  BUFSIZE 256

int main()
{
    Engine *ep;
    mxArray *T = NULL, *result = NULL;
    char buffer[BUFSIZE+1];
    double time[10] = { 0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0 };

  /*  调用 engOpen 语句启动 MATLAB 引擎 */
    if (!(ep = engOpen("\0"))) {
        fprintf(stderr, "\nCan't start MATLAB engine\n");
        return EXIT_FAILURE;
    }

    /*  PART I  这部分为 MATLAB 接收程序传来的数据，然后对其分析最后绘图 */

    /* 创建临时变量 */
    T = mxCreateDoubleMatrix(1, 10, mxREAL);
    memcpy((void *)mxGetPr(T), (void *)time, sizeof(time));

    /* 变量 T 放到 MATLAB 工作空间  */
    engPutVariable(ep, "T", T);
```

```
/* 计算 distance = (1/2)g.*t.^2 的值，g 为重力加速度，t 为时间 */
engEvalString(ep, "D = .5.*(-9.8).*T.^2;");

/* 结果绘图 */
engEvalString(ep, "plot(T,D);");
engEvalString(ep, "title('Position vs. Time for a falling object');");
engEvalString(ep, "xlabel('Time (seconds)');");
engEvalString(ep, "ylabel('Position (meters)');");

/* fgetc()确保结果图像停留时间足够长，如果太短则“看”不到图像 */
printf("Hit return to continue\n\n");
fgetc(stdin);

/* 第一部分完毕，释放内存，关闭 MATLAB 引擎 */
printf("Done for Part I.\n");
mxDestroyArray(T);
engEvalString(ep, "close;");

/*  PART II  这部分要求用户输入一个字符串，MATLAB 执行这个字符串，
 * 然后创建变量 X，并识别 X 的数据类型 */

/* 使用 engOutputBuffer 函数可以截获 MATLAB 的输出，
 * 此字符串必须以 NULL 结尾 */

buffer[BUFSIZE] = '\0';
engOutputBuffer(ep, buffer, BUFSIZE);
while (result == NULL) {
    char str[BUFSIZE+1];

    /* 获取用户的输入 */
    printf("Enter a MATLAB command to evaluate.  This command should\n");
    printf("create a variable X.  This program will then determine\n");
    printf("what kind of variable you created.\n");
    printf("For example: X = 1:5\n");
    printf(">> ");

    fgets(str, BUFSIZE, stdin);

    /* 执行用户输入的字符串 */
    engEvalString(ep, str);

    /* 在命令行中返回输出，前两个字符总是“>>” */
    printf("%s", buffer+2);

    /* 获取计算结果 */
    printf("\nRetrieving X...\n");
    if ((result = engGetVariable(ep,"X")) == NULL)
      printf("Oops! You didn't create a variable X.\n\n");
    else {
```

```
        printf("X is class %s\t\n", mxGetClassName(result));
        }
    }

    /* 完毕，释放内存，关闭 MATLAB 引擎，退出  */
    printf("Done!\n");
    mxDestroyArray(result);
    engClose(ep);

    return EXIT_SUCCESS;
}
```

用下面的格式编译此文件（具体细节见 10.4.3 节）：

```
mex -f lccengmatopts.bat engdemo.c
```

编译成功后，会生成一个可执行文件 engdemo.exe。在 Windows 下或 MATLAB 中就可以直接执行了。在 Windows 的【开始】→【运行】中输入“engdemo”可以执行 engdemo，在 MATLAB 控制窗口中，输入：

```
!engdemo
```

也可以执行此程序。但是笔者建议关闭 MATLAB 程序后，在 Windows 的 cmd 命令行中执行此程序，这样能更好观察到 C 程序调用 MATLAB 引擎。

执行此程序后，系统会启动 MATLAB 引擎和 MATLAB 控制窗口，用户可以在此控制窗口中输入 MATLAB 命令运行。

该程序第一部分运行结果为结果绘图，如图 10-11 所示。

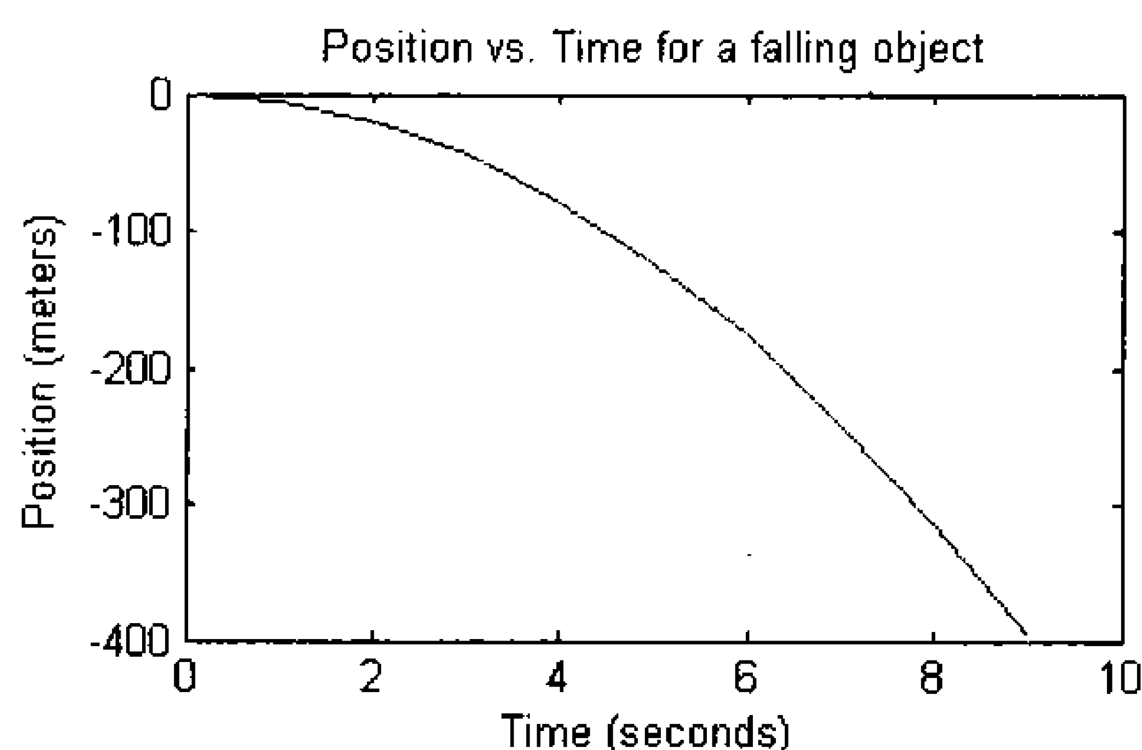


图 10-11　C 语言 engdemo 的执行结果

同时，第一部分执行完毕后（程序会停顿一段时间，保证图像正确显示），会显示以下信息：

```
Press Return to continue
```

按【Enter】键后，显示下列信息：

```
Done for Part I.
Enter a MATLAB command to evaluate.  This command should
create a variable X.  This program will then determine
what kind of variable you created.
For example: X = 1:5
```

接着输入下面指令（必须以大写的“X”开头）：

```
X = magic(3)
```

结果如下：

```
X =
        8     1     6
        3     5     7
        4     9     2

Retrieving X...
X is class double
Done!
```

到此程序结束，会自动关闭 MATLAB 引擎和绘图界面。

2．Fortran 程序调用 MATLAB 引擎

例 10.16　演示标准独立的 Fortran 程序中调用 MATLAB 引擎。

源文件为 fengdemo.F，与 engdemo.c 同目录，内容如下：

fengdemo.F

```
#include "fintrf.h"
      program main
      mwpointer engOpen, engGetVariable, mxCreateDoubleMatrix
      mwpointer mxGetPr
      mwpointer ep, T, D
      double precision time(10), dist(10)
      integer engPutVariable, engEvalString, engClose
      integer temp, status
      data time / 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0 /
C
      ep = engOpen('MATLAB ')
C
      if (ep .eq. 0) then
         write(6,*) 'Can''t start MATLAB engine'
         stop
      endif
C
      T = mxCreateDoubleMatrix(1, 10, 0)
      call mxCopyReal8ToPtr(time, mxGetPr(T), 10)
C
C
C       将变量 T 传送到 MATLAB 工作空间
C
      status = engPutVariable(ep, 'T', T)
C
      if (status .ne. 0) then
         write(6,*) 'engPutVariable failed'
         stop
      endif
C
      if (engEvalString(ep, 'D = .5.*(-9.8).*T.^2;') .ne. 0) then
         write(6,*) 'engEvalString failed'
         stop
      endif
```

```
C
C       绘图
C
        if (engEvalString(ep, 'plot(T,D);') .ne. 0) then
          write(6,*) 'engEvalString failed'
            stop
        endif

        if (engEvalString(ep, 'title(''Position vs. Time'')') .ne. 0) then
              write(6,*) 'engEvalString failed'
              stop
        endif

        if (engEvalString(ep, 'xlabel(''Time (seconds)'')') .ne. 0) then
              write(6,*) 'engEvalString failed'
              stop
        endif
        if (engEvalString(ep, 'ylabel(''Position (meters)'')') .ne. 0)then
              write(6,*) 'engEvalString failed'
              stop
        endif
C
C       print *, 'Type 0 <return> to Exit'
        print *, 'Type 1 <return> to continue'

        read(*,*) temp
C
        if (temp.eq.0) then
            print *, 'EXIT!'
            status = engClose(ep)

            if (status .ne. 0) then
                write(6,*) 'engClose failed'
            endif

            stop
        end if
C
        if (engEvalString(ep, 'close;') .ne. 0) then
            write(6,*) 'engEvalString failed'
            stop
        endif
C
        D = engGetVariable(ep, 'D')
        call mxCopyPtrToReal8(mxGetPr(D), dist, 10)
        print *, 'MATLAB computed the following distances:'
        print *, '   time(s)   distance(m)'
        do 10 i=1,10
            print 20, time(i), dist(i)
20          format(' ', G10.3, G10.3)
```

```
 10     continue
C
C       call mxDestroyArray(T)
        call mxDestroyArray(D)
        status = engClose(ep)
C
        if (status .ne. 0) then
           write(6,*) 'engClose failed'
           stop
        endif
C
        stop
        end
```

与 engdemo.c 类似，编译上述文件后，生成独立的可执行程序 fengdemo.exe。此程序的绘图结果和 C 语言版本的基本相同。类似地，在控制窗口也显示下列信息：

```
Type 0 <return> to Exit
Type 1 <return> to continue
```

输入 1 后按回车键，程序继续执行：

```
1
 MATLAB computed the following distances:
    time(s)   distance(m)
    1.00       -4.90
    2.00       -19.6
    3.00       -44.1
    4.00       -78.4
    5.00       -123.
    6.00       -176.
    7.00       -240.
    8.00       -314.
    9.00       -397.
    10.0       -490.
```

至此，此程序执行完毕，释放内存，关闭 MATLAB 引擎。

10.4.4 Visual C++建立与调试计算引擎程序

10.4.3 节是在 MATLAB 的编辑器中编辑 C 语言的引擎程序并在 MATLAB 环境中编译成可执行程序。本节介绍在 Visual C++中的 MATLAB 计算引擎程序的建立与调试。

基于 Visual C++和 MATLAB 混合编程是很多熟悉 Visual C++编程而又需要进行科学计算和数据仿真的科研人员常用的一种方式，其中最简单也最直接的方法就是调用 MATLAB 引擎。这样，Visual C++开发环境功能强大、开发的程序执行速度快弥补了 MATLAB 由于解释性语言而执行速度慢等缺点，而调用 MATLAB 计算引擎则弥补了 Visual C++在科学计算方面函数库显得不够丰富和显示图形不方便的缺点。

例 10.17 设计一程序，在 C/C++语言中调用 mesh 函数来绘制高斯矩阵的曲面。

解题步骤如下：

（1）首先在 Visual C++中建立 Win32 控制台程序，工程命名为 meshdemo。在下一步

中的“应用程序设置”中，选中“空项目”，其他保持默认选项。

（2）编写程序代码。

新建一个C++源文件，并保存为meshdemo.cpp，然后将此文件添加到meshdemo工程中作为其源文件，其内容如下：

meshdemo.cpp

```
#include <stdio.h>
#include "engine.h"
void main()
{
    Engine *ep;          //定义引擎指针
    int status = 0;

    ep = engOpen(NULL);    // 打开计算引擎
    if( ep == (Engine *)NULL ){
        printf("错误：无法打开 MATLAB 计算引擎\n" );
        exit(-1);       //若打开失败，退出程序
    }

    engEvalString(ep,"mesh(peaks);");    // 执行 MATLAB 指令
    getchar();

    status = engClose(ep);  // 关闭 MATLAB 计算引擎
    if(status != 0){
        printf("无法正常关闭 MATLAB 计算引擎\n");
        exit(-1);
    }
}
```

（3）设置工程属性，配置编译器。

由于此C++文件包含了调用MATLAB引擎的函数，所以必须包含“engin.h”头文件，另外，要成功编译MATLAB引擎程序，还需对Visual C++的编译器作一些必要设置。编译器设置与10.2.4节“Visual C++中建立MEX文件与调试”中的设置相同，其中“链接器”的“附加依赖项”改为libmx.lib和libeng.lib。

（4）编译和生成meshdemo.exe可执行文件。

（5）运行。在Visual C++中运行此生成的程序，得到如图10-12所示的结果。

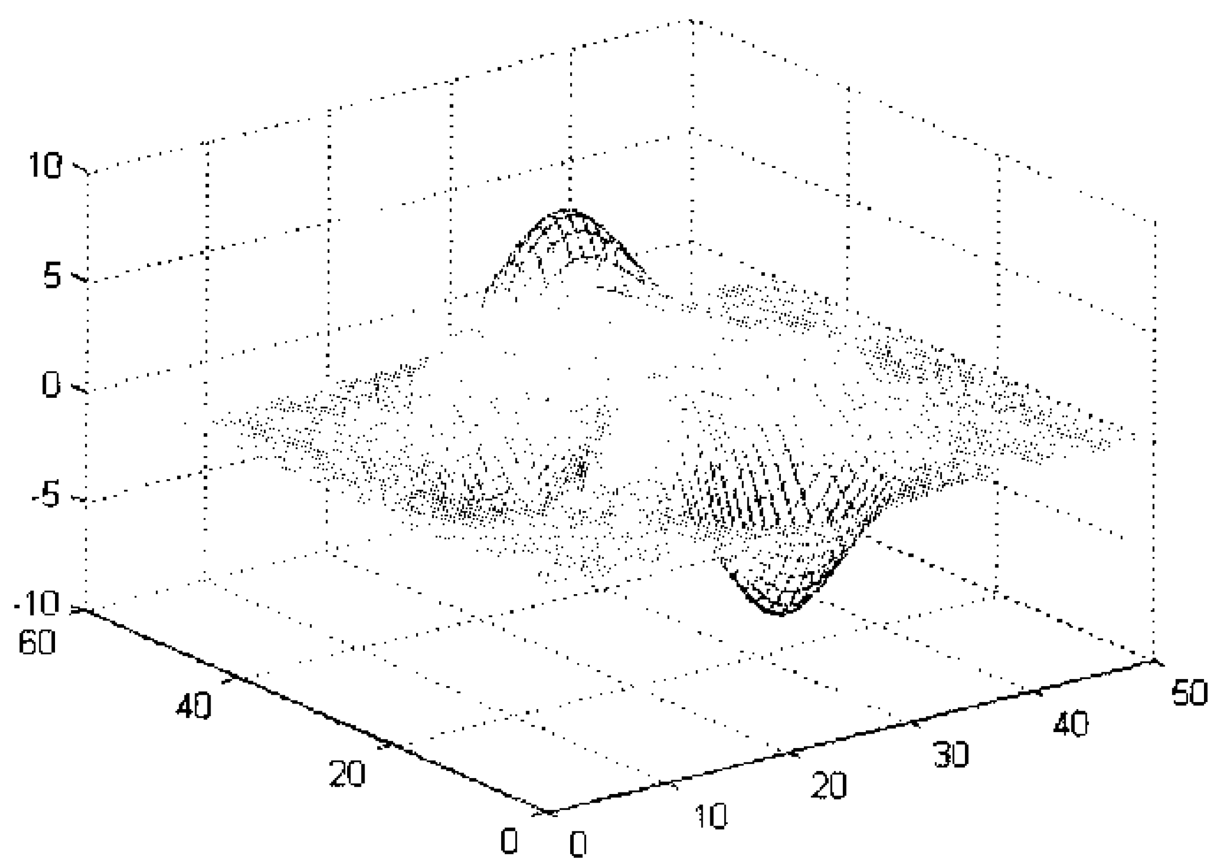


图10-12 meshdemo的执行结果

在开始 Visual C++中创建 MATLAB 引擎程序时，可能会出现编译错误或链接错误，这一般是编译器设置不正确。另外，MATLAB 的引擎程序可以在 Visual C++环境中像其他程序一样调试。

10.4.5 工程实例分析

10.4.4 节中介绍了一个简单的 MATLAB 引擎编程例子，仅调用了一条 MATLAB 语句。本节介绍一个略为复杂的例子。

1．MATLAB 数据类型 mxArray 的操作

实际应用中可能会调用 MATLAB 处理复杂的数据，如大型计算和绘图，这时需要 Visual C++与 MATLAB 进行数据交互，而二者的数据类型有很大不同，必须将二者的数据类型进行必要的转换。

在 Visual C++中，所有和 MATLAB 的数据交互都是通过 mxArray 来实现的，而且 MATLAB 引擎函数中，所有与变量有关的数据类型都是 mxArray 类型。数据结构 mxArray 以及大量的以 mx 开头的函数，广泛用于 MATLAB 引擎程序和 MATLAB 的 C 数学库中。数据类型 mxArray 在头文件 matrix.h 中定义，可以打开此文件来查看 mxArray 的具体数据结构。mxArray 是一种很复杂的数据结构，与 MATLAB 中的 array 相对应，我们只需熟悉 MATLAB 的 array 类型和几个常用的 mxArray 函数即可。

1）创建和清除 mxArray 型数据

MATLAB 有很多种变量类型，对应于每种类型，基本上都有一个函数用于创建，但它们都有相同的数据结构，就是 mxArray。

数组的建立采用 mxCreateXXX 形式的函数，其中 XXX 是具体的数据类型。如新建一个 double 类型数组，可用函数 mxCreateDoubleMatrix，函数形式如下：

```
mxArray *mxCreateDoubleMatrix(int m, int n, mxComplexity ComplexFlag);
```

其中参数 m 和 n 为矩阵的函数和列数。ComplexFlag 为常数，用来区分矩阵中元素是实数还是复数，取值分别为 mxREAL 和 mxCOMPLEX。

例如，创建一个 3 行 5 列的二维实数数组，可用如下语句：

```
mxArray *T = mxCreateDoubleMatrix(3, 5, mxREAL);
```

对应地，要删除一个数组使用 mxDestroyArray 函数可用如下语句：

```
void mxDestroyArray(mxArray *array_ptr);
```

参数 array_ptr 为要删除的数组指针。例如，要删除上面创建的数组 T，可用如下语句：

```
mxDestroyArray(T);
```

类似地，创建函数可用如下语句：

```
mxArray *mxCreateString(const char *str);
```

功能是创建一个字符串类型并初始化为 str 字符串。

一般的在 Visual C++与 MATLAB 交互中，以上几种类型就够了，其他类型数组的创建这里不再介绍。

2）管理 mxArray 数据类型

（1）管理 mxArray 数据大小

要获得 mxArray 数组每一维上元素的个数，可以用 mxGetM 和 mxGetN 函数。其中，

mxGetM 用来获得数组第一维的元素个数，对于矩阵来说就是行数。例如：

```
int mxGetM(const mxArray *array_ptr);
```

功能是返回 array_ptr 对应数组第一维的元素个数，即第一维的行数。

```
int mxGetN(const mxArray *array_ptr);
```

功能是返回 array_ptr 对应数组其他维的元素个数，对于矩阵来说是列数。对于多维数组来说是从第二维到最后一维的各维元素个数的乘积。

要获得某一特定维的元素个数，则要用函数：

```
const int *mxGetDimensions(const mxArray *array_ptr);
```

该函数返回 array_ptr 各维的元素个数保存在一个 int 数组中返回。对于常用的矩阵来说，用 mxGetM 和 mxGetN 两个函数就可以了。

另外，还可以通过 mxGetNumberOfDimensions 来获得数组的总维数，用 mxSetM、mxSetN 设置矩阵的行数和列数，函数说明如下：

```
int mxGetNumberOfDimensions(const mxArray *array_ptr);
```

功能为返回数组的维数。

```
void mxSetM(mxArray *array_ptr, int m);
```

功能为设置 array_ptr 所指的数组为 *m* 行。

```
void mxSetN(mxArray *array_ptr, int n);
```

功能为设置 array_ptr 所指的数组为 *n* 列。

（2）判断 mxArray 数组类型

在对 mxArray 类型的变量进行操作之前，可以验证以下数组的数据类型，如是否为 double 数组、整数、字符串和逻辑值等，以及是否为某种结构、类、或者是特殊类型，如是否为空数组，是否为 inf 和 NaN 等。常见的判断函数有：

```
bool mxIsDouble(const mxArray *array_ptr);
bool mxIsComplex(const mxArray *array_ptr);
bool mxIsChar(const mxArray *array_ptr);
bool mxIsEmpty(const mxArray *array_ptr);
bool mxIsInf(double value);
```

这些函数的含义比较明显，便于识记和使用。

（3）管理 mxArray 数组的数据

对于常用的 double 类型的数组，可以用 mxGetPr 和 mxGetPi 两个函数分别获得其实部和虚部的数据指针，这两个函数的声明如下：

```
double *mxGetPr(const mxArray *array_ptr);
```

功能为返回数组 array_ptr 的实部指针。

```
double *mxGetPi(const mxArray *array_ptr);
```

功能为返回数组 array_ptr 的虚部指针。

这样，就可以通过获得的指针对 mxArray 类型的数组中的数据进行读写操作。例如，可以用函数 engGetVariable 从 MATLAB 工作空间读入 mxArray 类型的数组，然后用 mxGetPr 和 mxGetPi 获得数据指针，对并其中的数据进行处理，最后调用 engPutVariable 函数将修改后的数组重新写入到 MATLAB 工作空间。

2. 程序实例

下面通过几个实例具体演示利用 Visual C++调用 MATLAB 引擎，通过这些例子，会初

步掌握 MATLAB 引擎编程，要灵活运用还要多加练习，在编程和调试中积累技巧。

1）mxArray 数组创建、绘图

例 10.18　由 $y=\sin(x)\pm\log(x)$绘图。

本例演示如何利用 Visual C++创建 MATLAB 的 mxArray 类型，然后利用这些数据绘图。在 Visual C++中新建控制台工程，编写代码如下：

sinlog.cpp

```
#include <iostream>
#include <math.h>
#include "engine.h"
using namespace std;

void main()
{
    const int N = 50;
    double x[N],y[N];
    int j = 1;

    // 计算数组 x 和 y
    for (int i=0; i<N; i++)
        {
            x[i] = (i+1);
            y[i] = sin(x[i]) + j * log(x[i]);
            j *= -1;
        }
    Engine *ep;
    if (!(ep=engOpen(NULL)))
        {
            cout <<"Can't start MATLAB engine!" <<endl;
            exit(1);
        }

    // 定义 mxArray，为 1 行，N 列的实数数组。
    mxArray *xx = mxCreateDoubleMatrix(1,N, mxREAL);
    mxArray *yy = mxCreateDoubleMatrix(1,N, mxREAL);

    // 将数组 x 和 y 分别复制到 mxarray 数组 xx、yy 中。
    memcpy(mxGetPr(xx), x, N*sizeof(double));
    memcpy(mxGetPr(yy), y, N*sizeof(double));
    engPutVariable(ep, "xx",xx);
    engPutVariable(ep, "yy",yy);

    // 向 MATLAB 引擎发送画图命令
    engEvalString(ep, "plot(xx, yy); ");
    engEvalString(ep, "title(' y  =  sin(x) ±log(x)');");
    engEvalString(ep, "xlabel('x');");
    engEvalString(ep, "ylabel('y');");
```

```
    // 销毁 mxArray 数组 xx 和 yy
    mxDestroyArray(xx);
    mxDestroyArray(yy);
    cout <<"Press any key to exit!" <<endl;
    cin.get();

    // 关闭 MATLAB 引擎。
    engClose(ep);
}
```

编译器按照例 10.17 设置后，编译生成可执行程序并运行程序，得到的结果如图 10-13 所示。

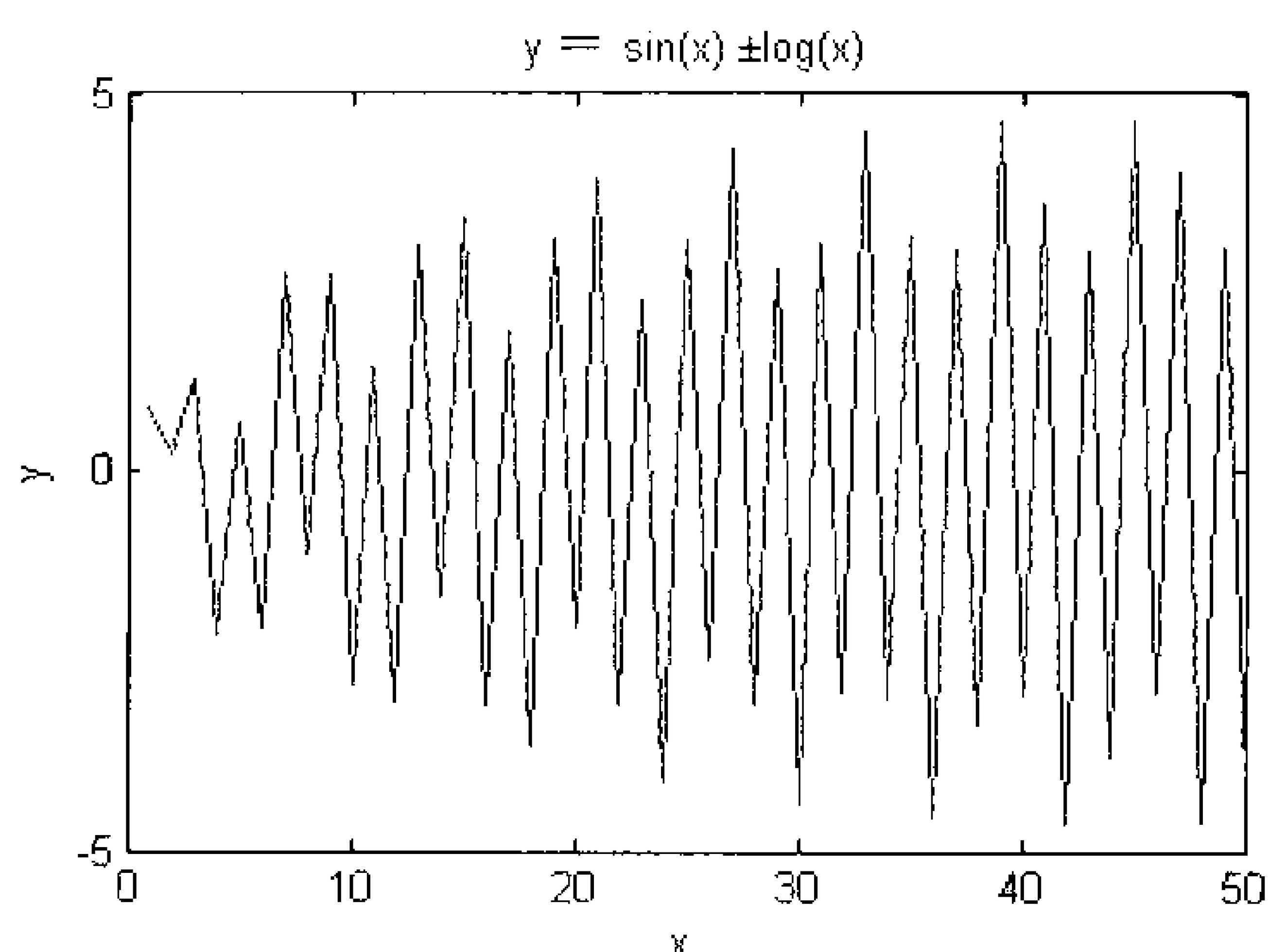


图 10-13　函数 $y=\sin(x)\pm\log(x)$的图像

2）利用引擎数学计算

例 10.19　在 C/C++语言中调用 MATLAB 计算魔方阵，并将结果显示到屏幕。

在 Visual C++中建立控制台工程，名为 magic，创建源文件 magic.cpp，内容如下：

magic.cpp

```
#include "engine.h"
#include <stdio.h>
void main()
{
    Engine *ep;
    double *p;
    mxArray *equation;
    int i = 0,j=0;
    int status = 0;
    // 打开计算引擎
    ep = engOpen(NULL);
    if( ep == (Engine *)NULL ){
        printf("错误：无法打开 MATLAB 计算引擎\n" );
        exit(-1);
    }
    // 执行 MATLAB 指令
    engEvalString(ep,"A = magic(5);");
    equation=engGetVariable(ep,"A");
```

```
        p=mxGetPr(equation);
        printf("\nMATLAB 中计算 magic(5)\n");
        for(i=0;i<5;i++)
        {
            for(j=0;j<5;j++)
                printf("%8.0lf",*(p+i+j*5));
            printf("\n");
        }
        // 关闭 MATLAB 计算引擎
        status = engClose(ep);
        if(status != 0){
            printf("无法正常关闭 MATLAB 计算引擎\n");
            exit(-1);
        }
    }
```

按照例 10.17 设置编译器，然后编译和运行，得到的结果如图 10-14 所示。

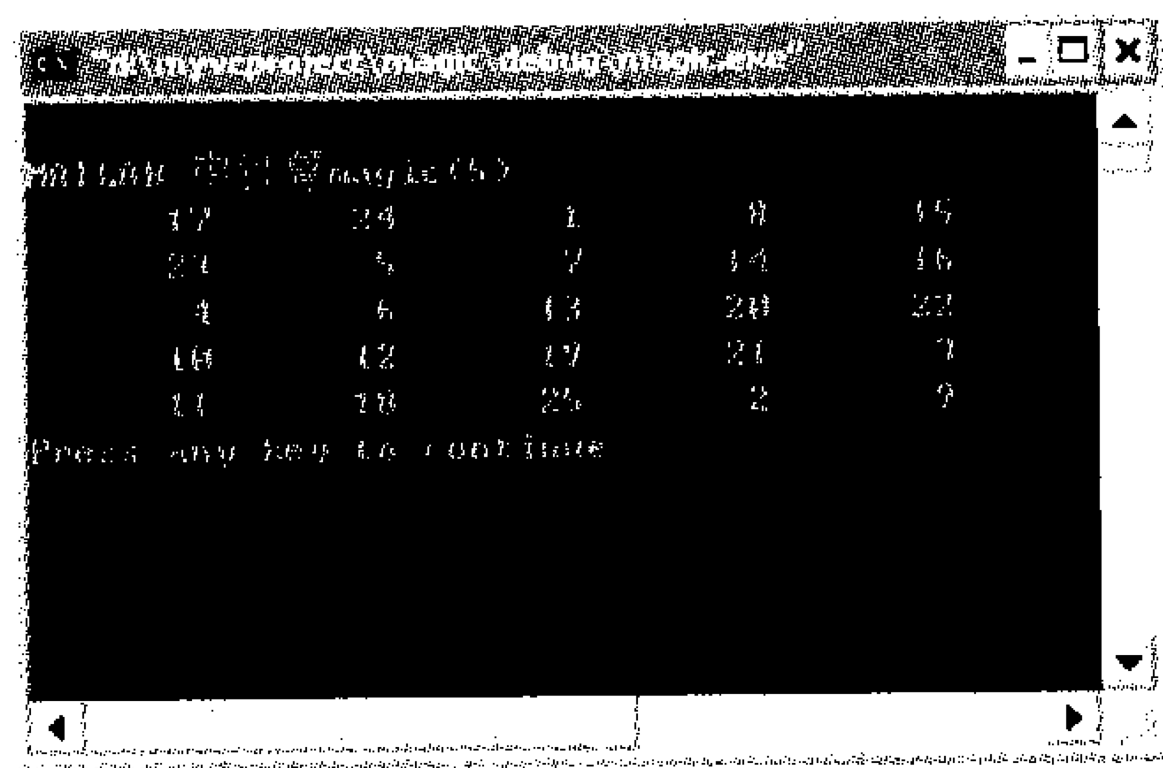


图 10-14　magic 运行显示的 5 阶魔方

3）解线性方程组

例 10.20　在 C/C++语言中调用 MATLAB 求解下面的线性方程组。

$$\begin{cases} 3x_1 - x_2 + 3x_3 = 2 \\ 4x_1 + 2x_2 - 6x_3 = 5 \\ 5x_1 - 3x_2 + 9x_3 = 8 \end{cases}$$

在 Visual C++中创建控制台工程，建立编辑源文件 linear_equation.cpp，内容如下：

linear_equation.cpp

```
#include <stdio.h>
#include <string.h>
#include "engine.h"
void main()
{
    Engine *ep;
    int status = 0;
    // 启动 MATLAB 引擎运行 MATLAB 指令
    ep = engOpen(NULL);
    double A[]={3, 4, 5, 1, 2, 3, 3, 6, 9},B[]={2,5,8};
    double *pa,*pb,*pc;
    mxArray *x,*y,*z;
```

```
        x=mxCreateDoubleMatrix(3,3,mxREAL);
        y=mxCreateDoubleMatrix(3,1,mxREAL);
        pa=mxGetPr(x);
        pb=mxGetPr(y);
        memcpy(pa,A,9*sizeof(double));
        memcpy(pb,B,3*sizeof(double));
        engPutVariable(ep,"A",x);
        engPutVariable(ep,"B",y);
        engEvalString(ep,"C=A\\B;");
        z=engGetVariable(ep,"C");
        pc=mxGetPr(z);
        printf("\n 方程组 AX=B 的结果为：\n");
        for(int i=0;i<3;i++)
        {
            printf("%8.0lf\n",*(pc+i));
        }
        status = engClose(ep);
    }
```

按照例 10.17 设置编译器，然后编译和运行，得到方程组的解如图 10-15 所示。

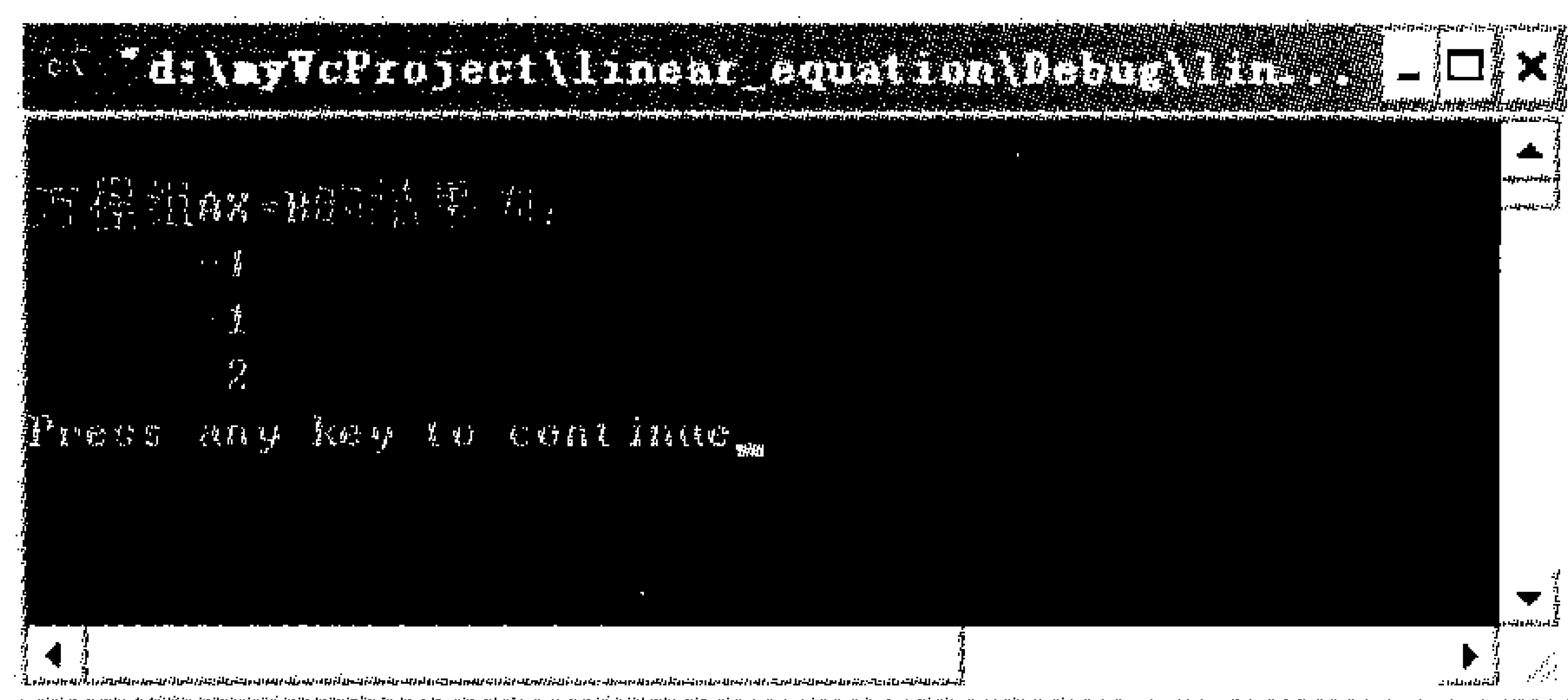


图 10-15　线性方程组的解

大多时候，程序员可以利用 MATLAB 强大的数据读写、显示能力和 Visual C++编程的高效率。如在 MATLAB 中要读入一幅任意格式的图像均只需一条命令：

```
pic = imread('mypic.jp');
```

图像数据矩阵便存放在了二维数组 pic 中，可以通过 Visual C++读入该数组进行相关处理再调用 MATLAB 显示，这种混合编程方式能大大提高工作效率。

当然，利用 Visual C++编译的 MATLAB 引擎程序，运行环境中还必须有 MATLAB 的支持，如果要编译完全脱离 MATLAB 的程序，可采用其他方式，如利用第三方 Matcom 程序编译独立的可执行程序等。

10.5　MATLAB COM 和 DDE 编程

组件对象模型（Component Object Model，COM）提供了一个可将可重用的软件集成到应用程序中的框架。MATLAB 支持 COM 技术，大大简化了应用程序开发。MATLAB 提供了使 MATLAB 与其他 Windows 应用程序相互访问的函数。这些函数使用动态数据交换

（Dynamic Data Exchange，DDE），DDE 允许 Windows 应用程序之间以交换数据形式通信。

本节介绍了 MATLAB COM 编程，包括客户端编程和作为自动化服务器编程，最后简要介绍了 MATLAB DDE 编程，包括服务器和客户端编程。

10.5.1 MATLAB COM 概述

组件对象模型提供了将可重用的二进制软件组件集成为一个应用程序的框架。MATLAB 支持 COM 技术，所以大大简化了应用程序开发：因为组件是以编译后的代码的形式，源码可以是任意支持 COM 的编程语言编写。使用组件的应用程序升级变得简单，因为升级组件时不必重新编译整个应用程序；另外，组件的位置对于应用程序来说是透明的，所以不必修改应用程序即可重新载入组件。使用 COM 技术，开发者和用户可以选择不同厂商生产的组件集成到一起，由此来创建一个应用程序。

使用 COM 技术，开发者和终端用户可以选择不同厂商生产的组件并集成为一个完整的应用程序。如一个单独应用程序需要数据库访问、数学分析以及商业质量图形显示等。若使用 COM，开发者就可以选择一个厂商提供的数据库访问组件， 而选择另外厂商提供的商业图形组件，然后将这两个组件集成到第三个厂商提供的数学分析开发包组件中。所以开发者可以像搭积木一样创建自己的应用程序，而积木都由不同厂商提供，可见使用 COM 编程无疑使用了现有资源的重用性，简化了开发过程，而且结构清晰，便于调试修改。

1．概念和术语

下面具体了解一下 COM 编程的一些概念和术语，这对 COM 编程是非常必要的。

1）程序标识符（Programmatic Identifiers）

在 MATLAB 内部创建 COM 对象时，必须由程序标识符或 ProgID 来引用这个对象。组件的 ProgID 是由其厂商定义的，用来区分其他组件的唯一字符串。在组件厂商的文档中可以找到其 ProgID。

2）自动化（Automation）、定制（Custom）和双向（Dual）服务器类型

COM 服务器的类型取决于其实现的接口的类型，类型有下面几种。

- 自动化：支持 OLE 自动化的标准，自动化服务器基于 IDispatch 接口。
- 定制：自定义的接口，允许客户端更直接、更快速地访问服务器对象的属性和方法。自定义接口基于 IUnknown 接口。
- 双向：由自动化和定制服务器结合而成。

COM 对象的基本接口类型有以下几种。

- IDispatch：工业标准接口，是一个较小的函数集，实现获取 COM 对象信息、访问 COM 对象属性和方法等功能。
- IUnknown：工业标准接口，所有的 COM 对象都要求有此接口，其他接口都是派生于 IUnknown。

3）进程内（In-Process）和进程外（Out-of-Process）服务器

配置服务器有以下 3 种方式，MATLAB 对此全部支持。

（1）进程内服务器

若组件实现形式为动态链接库（DLL）或 ActiveX 控件，则运行时与客户端应用程序一个进程，共享地址空间。这使得客户端和服务器之间的通信简单而快速。

（2）本地进程外服务器

多组件为可执行（.exe）文件，则运行时作为一个单独的进程，但客户端程序和服务器程序在同一台机器上。由于进程间通信的消耗，所以这样的配置略慢。

（3）远程进程外

这是另一种进程外服务器类型，不同的是服务器和客户端在不同的机器上，由网络来通信。显然由于除了进程间通信，还有网络延时，所以这种配置方式最为耗时。

2．注册控件和服务器

大部分控件和服务器在程序安装时是自动注册的。当有新的控件，如.ocx、.dll 类型或其他类型的控件或服务器，则必须手动来注册。使用 DOS 命令“regsvr32”可以在 Windows 下实现这个功能。

在 DOS 命令行中，先用“cd”命令进入要注册文件的目录，然后输入下面命令

```
regsvr32 filename.ocx
```

即可对 filename.ocx 组件进行注册。如注册 MATLAB 例子中常用的 mwsamp2.ocx 文件，输入下面命令：

```
regsvr32 mwsamp2.ocx
```

使用 regsvr32 命令后，以 MATLAB 的 mwsamp 控件为例，可使用下列方法来验证组件是否成功注册：

- 用 Visual Studio .NET 2005 的 ActiveX control test container 工具，单击【编辑】菜单，然后选择【插入新控件】→【MwSamp Control】命令，若能成功地插入此控件，则说明此控件已成功注册了。此方法仅对控件可用。
- 在 DOS 命令行输入“regedit”，使用“注册编辑器”。利用【编辑】菜单的【查找】功能查找出要检测的控件或服务器，类似下面的结构:

  ```
  HKEY_CLASSES_ROOT/Program ID
  ```
- 用 Visual Studio .NET 的“OLE/COM Object Viewer”工具查看。

10.5.2 MATLAB COM 客户端编程

利用 COM 技术，MATLAB 直接集成其他 COM 组件来完成软件开发，而不必重新编写相同功能的模块。下面介绍操作控件的函数，再结合实例介绍 MATLAB 的 COM 客户端编程。

1．创建控件或服务器

有两个 MATLAB 的函数可以创建 COM 控件或服务器：actxcontrol 和 actxserver。每个函数都返回对象接口的句柄，利用此句柄可以访问对象的属性、方法、事件以及提供的其他接口。下面具体介绍这两个函数。

1）actxcontrol 函数

功能为在 MATLAB 的 figure 中创建一个控件，其调用方式有以下几种。

```
h = actxcontrol('progid')
```

```
h = actxcontrol('progid','param1',value1,...)
h = actxcontrol('progid', position)
h = actxcontrol('progid', position, fig_handle)
h = actxcontrol('progid',position,fig_handle,event_handler)
h = actxcontrol('progid',position,fig_handle,event_handler,'filename')
```

可以根据需要来选择上面的调用方式。其中参数 progid 为上面讲的 ProgID；param 为父窗口的句柄；position 为此控件显示的位置，用[x y width height]的形式表示，其中 x 和 y 为控件左下角相对于图形窗口左下角的偏移距离，单位为像素，而 width 和 height 为控件的宽度和高度，position 的默认值为[20 20 60 60]；fig_handle 为图形窗口的句柄，控件在此窗口中显示；event_handler 为事件处理程序句柄；filename 为 M 文件名，必要时 MATLAB 调用此 M 文件来响应事件。

例 10.21　MATLAB figure 中显示日历（Calendar）控件。

在 MATLAB 控制窗口中输入：

```
m = figure;
cal = actxcontrol('mscal.calendar',[0 0 300 300],m);
```

此时，cal 为 calendar 控件的句柄，可以由此句柄来访问其属性和方法等。执行上述指令后，MATLAB 打开一个 figure 窗口，并显示日历控件，如图 10-16 所示。

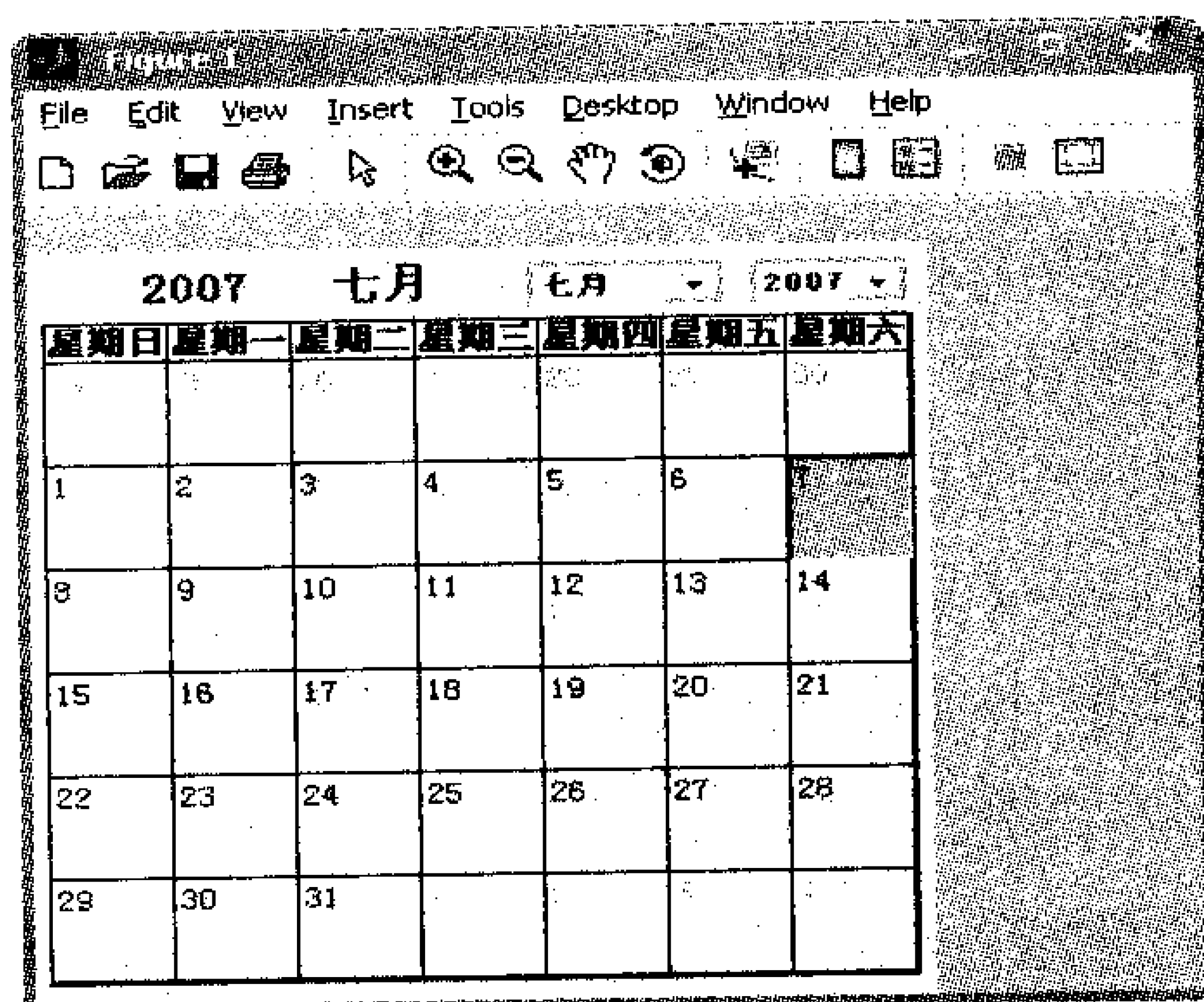


图 10-16　MATLAB 显示日历控件

创建控件后，若发现控件大小不合适，或显示位置不当使得控件显示不完全等不符合要求的情况，可以调用 move 函数更改其大小和显示位置。如对于此例，调用内容如下：

```
cal.move([70 120 400 350]);
```

则 calendar 控件的左下角会移到坐标（70，20）处，而且大小更改为 400×350。

2）actxserver 函数

功能为创建一个 OLE 自动化服务器或定制接口服务器，其调用格式如下：

```
h = actxserver('progid')
h = actxserver('progid', 'machine', 'machineName')
h = actxserver('progid', 'interface', 'interfaceName')
h = actxserver('progid','machine', 'machineName', 'interface', 'interfaceName')
h = actxserver('progid', machine)
```

可以根据具体需要选择调用方式。其中参数 machineName 为远程 machine 的名字字符串，interfaceName 为 COM 对象接口的名称字符串。

例 10.22　创建一个 OLE 自动化服务器，在此服务器中调用微软的 Excel。

在 MATLAB 控制窗口输入下面命令（或编辑成脚本 M 文件）：

```
e = actxserver ('Excel.Application')
% 使窗口可见
e.Visible = 1;
%  创建"eWorkBooks"接口
eWorkbooks = e.Workbooks
```

列出此接口所有的方法：

```
eWorkbooks.invoke
```

得到的结果如下：

```
ans =
          Add:  'handle Add(handle, [Optional]Variant)'
        Close:  'void Close(handle)'
         Item:  'handle Item(handle, Variant)'
         Open:  'handle Open(handle, string, [Optional]Variant)'
     OpenText:  'void OpenText(handle, string, [Optional]Variant)'
```

创建 workbook，名为 w：

```
w = eWorkbooks.Add
```

此时 Excel 中添加了一个 workbook，最后退出 Excel 并删除对象：

```
e.Quit;
e.delete;
```

3）图形界面创建控件对象

当不知道所要创建控件的 ProgID 时，可以用图形界面创建控件对象。相对于用函数创建控件对象，图形界面创建更简单，设置其属性更为方便。在 MATLAB 控制窗口调用 actxcontrolselect 函数即可打开一个图形用户界面，并显示出本机上安装的所有控件，如图 10-17 所示。当选中某个控件后，右上界面会有其图形预览，右下界面有其具体信息，包括此控件的 ProgID、位置，另外单击【Properties】按钮可以设置控件的属性，比如显示位置、父窗口句柄、响应事件，以及实现回调函数的 M 文件。

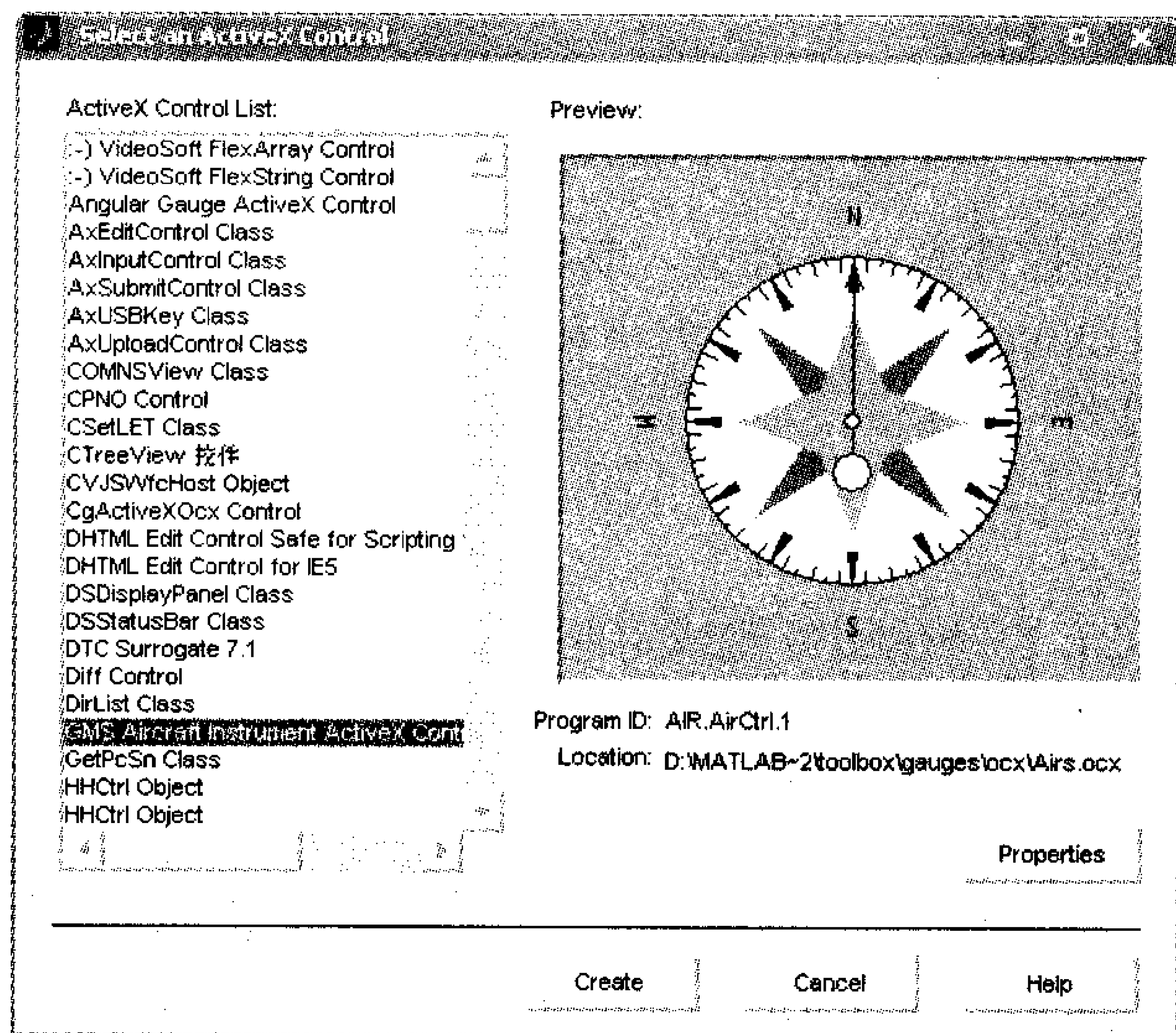


图 10-17　图形界面创建控件

另外，actxcontrollist 函数会返回本机上所有安装的控件信息，用此函数可以先查询本机安装控件的情况。

2．获取控件或服务器信息

对于某个具体对象，通常需要知道有哪些方法可以对其进行操作。用 get 和 events 函数可以完成此功能。如使用命令：

```
handle.get(handle) 或 get(handle,'PropertyName')
```

可以列出此对象的所有属性或某个指定属性的值。如对于 calendar 例子：

```
cal.get
```

得到下面的结果：

```
BackColor          :2147483663
Day                :7
DayFont            : [1x1 Interface.Microsoft_Forms_2.0_Object_Library.Font]
DayFontColor       : 0   % 节省篇幅，略去了后面的 18 个属性值一笔者添注
```

而命令 handle.methods 会列出句柄为 handle 的对象所支持的所有方法。如对于 calendar 例子：

```
cal.methods
```

得到下面的结果：

```
Methods for class COM.mscal_calendar:

AboutBox     NextYear        PreviousYear   constructorargs  get         move      send
NextDay      PreviousDay     Refresh        delete           interfaces  propedit  set
NextMonth    PreviousMonth   Today          deleteproperty   invoke      release
NextWeek     PreviousWeek    addproperty    events           load        save
```

用 set 命令可以对上面的属性值进行设置，另外，handle.events 列出对象所支持的所有事件，同样对于 calendar 例子：

```
cal.events
ans =
    Click = void Click()
    DblClick = void DblClick()
    KeyDown = void KeyDown(int16 KeyCode, int16 Shift)
    KeyPress = void KeyPress(int16 KeyAscii)
    KeyUp = void KeyUp(int16 KeyCode, int16 Shift)
    BeforeUpdate = void BeforeUpdate(int16 Cancel)
    AfterUpdate = void AfterUpdate()
    NewMonth = void NewMonth()
    NewYear = void NewYear()
```

10.5.3　MATLAB COM 自动化服务器编程

自动化（Automation）是一个 COM 协议，此协议允许一个应用程序（作为控制者）控制另外一个应用程序（作为服务器）中的对象。在 Windows 环境下，MATLAB 支持 COM 自动化服务器的特性与功能。任何可以配置成自动化控制者（Automation controller）的 Windows 程序都可以控制 MATLAB。比如，Microsoft Excel、Microsoft Access、Microsoft Project 以及 Visual Basic 和 Visual C++ 程序等都可以控制 MATLAB。

1．创建自动化服务器

10.5.2 节中讲过可以通过 actxserver 命令来创建自动化服务器，MATLAB 的 ProgID 为 matlab.application。如何创建自动化服务器取决于所使用的控制者，若控制者为 MATLAB 的应用程序，可以用下面的命令创建自动化服务器：

```
h = actxserver('matlab.application')
h =
    COM.matlab.application
```

MATLAB 自动化服务器有以下两种工作模式。

- 共享（Shared）：多个客户端应用程序连接到同一个 MATLAB 服务器，此服务器被所有客户端共享。
- 专有（Dedicated）：每个客户端创建各自专有的 MATLAB 服务器。

若使用 matlab.application 作为创建服务器时的 ProgID，则 MATLAB 默认被创建成共享模式的服务器。

使用 enableservice 函数可以获取当前 MATLAB 自动化服务器的状态值，函数返回一个逻辑值，1（true）表示 MATLAB 为一个自动化服务器，若返回 0（false）则相反。如输入以下命令来调用此函数：

```
enableservice('AutomationServer')
```

MATLAB 得到结果为 1，表示此时 MATLAB 正作为自动化服务器。

2．连接已有服务器

在需要使用 MATLAB 作为自动化服务器时，没必要每次都单独去创建一个新的 MATLAB 服务器。当 MATLAB 已经作为自动化服务器时，客户端就可以连接到此服务器并使用。调用 actxGetRunningServer 函数可以完成此功能，其中，actxGetRunningServer 调用方式如下：

```
h = actxGetRunningServer('progid')
```

如连接已有的 Excel 服务器，内容如下：

```
h = actxGetRunningServer('Excel.Application')
```

也可以使用 Visual Basic.NET，其命令 GetObject 也可以连接到已有的服务器，内容如下：

```
h = GetObject(, "matlab.application")
```

一定要按照上面的调用方式连接已有的 MATLAB 服务器：忽略第一个参数，第二个参数必须为“matlab.application”。

例 10.23 用 Visual Basic.NET 连接已有的 MATLAB 服务器，然后调用 plot 绘图。

代码如下：

```
Dim h As Object
h = GetObject(, "matlab.application")

h.Execute ("plot([0 18], [7 23])")
```

3．MATLAB 自动化服务器函数

MATLAB 提供了在 MATLAB 服务器中操作数据的一系列函数。下面具体介绍这些函数。

1）在服务器中执行命令

Execute 和 Feval 函数可以在服务器中执行 MATLAB 指令。

例如，用 Execute 函数调用 MATLAB 的 plot 函数绘图：

```
h.Execute ("plot([0 18], [7 23])")
```

用 Feval 函数调用 MATLAB 的 strcat 函数连接两个字符串：

```
a = h.Feval('strcat', 1, 'hello', ' world')
a =
    'hello world'
```

2）与服务器交换数据

MATLAB 提供了读写 MATLAB 服务器任何工作空间数据的函数。如表 10-6 所示介绍了 6 个读/写数据的函数及其用途。

表 10-6 读/写服务器数据的函数

函数	用途
Get/PutCharArray	读/写服务器中的字符串数组
Get/PutFullMatrix	读/写服务器中的矩阵
Get/PutWorkspaceData	读/写服务器中任意数据类型的数据

例 10.24 往 MATLAB 服务器基本工作空间的变量 str 写入字符串，然后在客户端读出此字符串。

在 MATLAB 控制窗口输入下面命令：

```
h = actxserver('matlab.application');
h.PutCharArray('str', 'base', 'He jests at scars that never felt a wound.');
```

读取字符串 str，检验写入是否成功：

```
S = h.GetCharArray('str', 'base')
S =
  He jests at scars that never felt a wound.
```

3）控制服务器窗口

控制服务器窗口的函数有以下两个。

- MaximizeCommandWindow：在 Windows 桌面上显示服务器窗口。
- MinimizeCommandWindow：最小化服务器窗口。

例如，创建一个 MATLAB 服务器，然后最小化窗口：

```
h = actxserver('matlab.application');
h.MinimizeCommandWindow;
```

4）actxcontrol 函数

当 MATLAB 服务器完成所有操作后，需要退出 MATLAB 模块并删除其进程，函数有以下两个。

- Quit：退出 MATLAB 模块。
- delete：删除 MATLAB 服务器进程。

注意这两个函数的大小写，因为 MATLAB 是区分大小写的。如退出 MATLAB：

```
h.Quit;
```

终止服务器进程：

```
h.delete;
```

10.5.4 MATLAB DDE 编程

动态数据交换（DDE）是一个微软开发的通信协议。该协议允许 Windows 中的应用程序之间交换数据和指令。交换数据的两个应用程序之间为 Server/Client 关系。发送交换数据请求的称为客户端，接受请求并提供数据的称为服务器。MATLAB 支持 DDE 协议，提供了 MATLAB 与其他 Windows 应用程序交换数据的函数。

1．DDE 概念与术语

交换数据的两个应用程序建立一个 DDE 对话，客户端发起请求并初始化 DDE 对话，必须初始化两个参数：服务名（Service Name）和主题（Topic）。这两个参数由服务器定义。

服务名为客户端要交换数据的应用程序的名称。每个可以作为 DDE 服务器的应用程序都有唯一的服务名。服务名通常是可执行文件名称去掉后缀，如 MATLAB 的服务名为 MATLAB，Word 的服务名为 WinWord，Excel 的服务名为 Excel。

主题为某次 DDE 对话的主要内容，主题的名字不区分大小写。MATLAB 的主题为 System 和 Engine，大多数应用程序支持 System 主题和至少一个其他主题。

当服务器接受客户端的请求后，就会建立一个 DDE 对话，每个对话的服务名和主题一起唯一确定了一个对话。建立连接后，服务名称和主题都不能改变。

在 DDE 对话过程中，客户端与服务器交换的数据单位称为“项目”（Item），而且服务器和客户端都可以改变项目。

DDE 使用 Windows 的剪贴板（Clipboard）格式作为服务器和客户端交换数据的格式。作为客户端，MATLAB 仅支持文本格式，作为服务器，MATLAB 不仅支持文本格式，还支持 Metafilepict 和 XLTable 格式。

2．MATLAB DDE 服务器编程

客户端应用程序可以将 MATLAB 作为一个 DDE 服务器来访问，MATLAB 提供了两种访问方式，具体哪种方式取决于客户端应用程序。两种方式如下。

- 若客户端应用程序提供了操作 DDE 对话的函数或宏，则可以直接调用这些函数或宏来访问。如微软的 Excel、Word 以及 Visual Basic.NET。
- 当没有提供直接访问的函数或宏时，就需要 MATLAB 的引擎库或直接访问 DDE。

图 10-18 说明了 MATLAB 作为一个服务器进行通信的方式。客户端应用程序的 DDE 函数与 MATLAB 的 DDE 服务器模块进行通信。客户端的 DDE 函数可以由客户端应用程序或 MATLAB 引擎库提供。

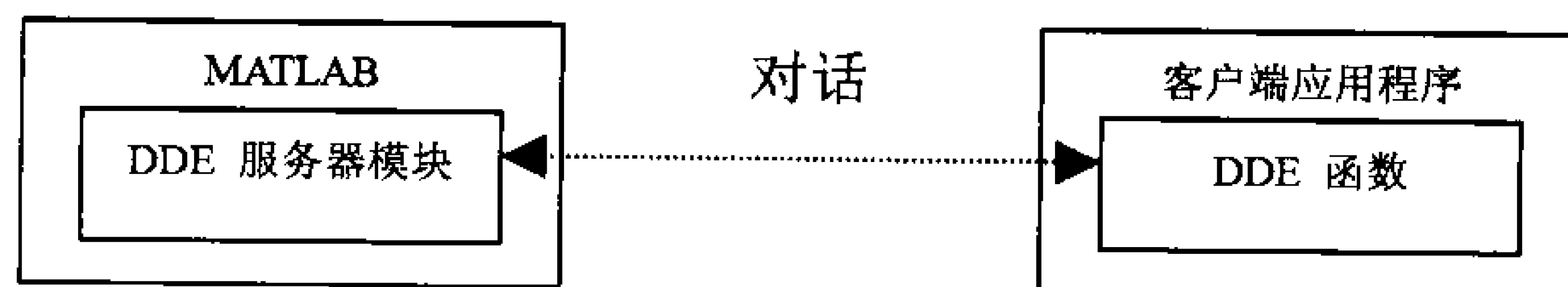


图 10-18　MATLAB 作为服务器时通信方式

1）*DDE 命名层次*

当作为服务器访问 MATLAB 时，必须指定服务名、主题和项目参数。如图 10-19 所示

介绍了 MATLAB DDE 编程中这 3 个参数的层次关系。

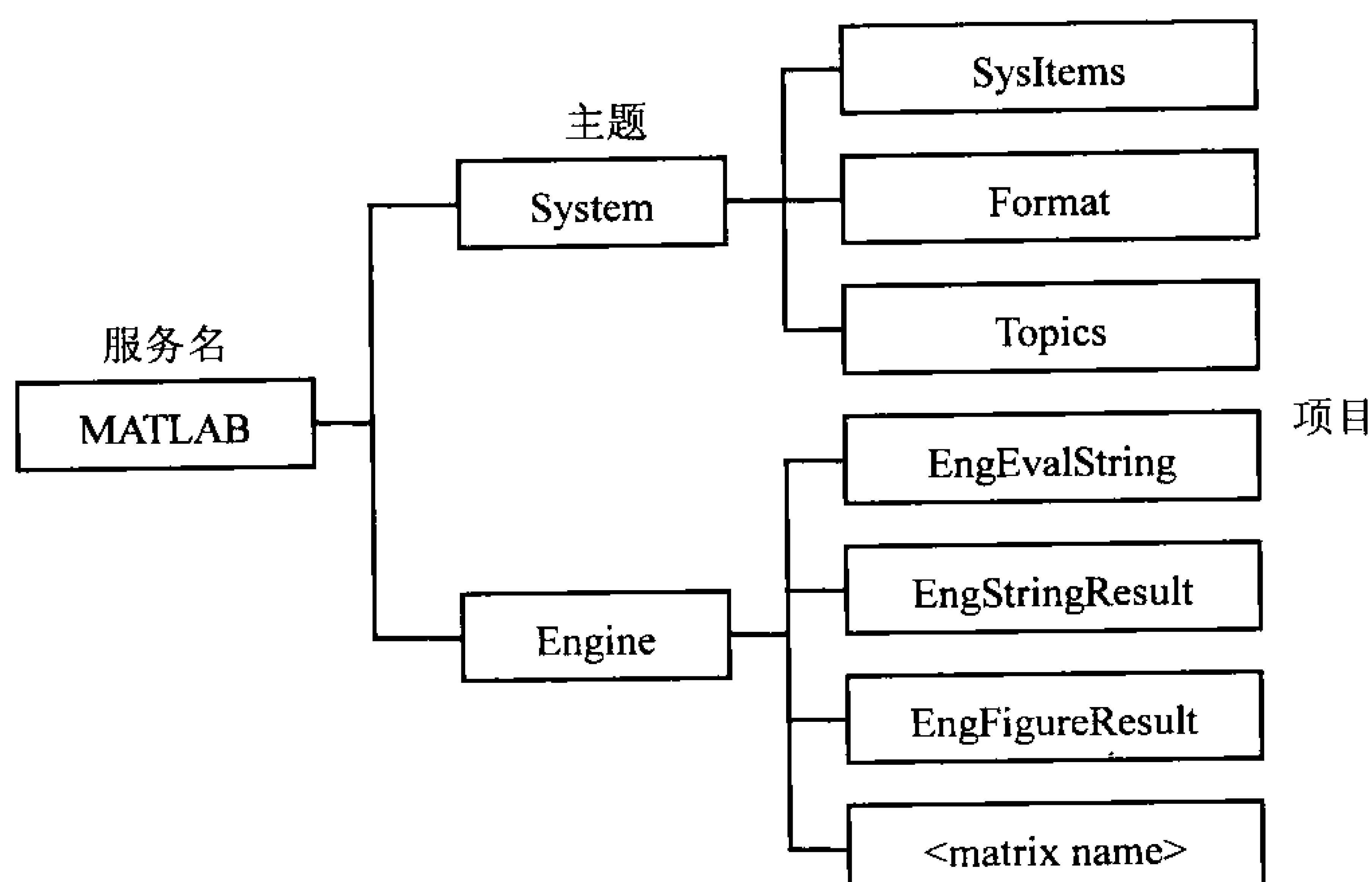


图 10-19　MATLAB DDE 命名层次

从图 10-19 中可见 MATLAB 仅有两个主题 System 和 Engine，而这两个主题支持的项目是不同的。System 支持 SysItems、Format 和 Topics 3 种类型项目，而 Engine 支持 EngEvalString、EngStringResult、EngFigureResult 以及<matrix name>。

2）*客户端与 MATLAB 交互*

MATLAB 的 Engine 主题支持 3 种操作：发送命令让 MATLAB 执行、请求 MATLAB 的数据以及向 MATLAB 发送数据。

用 execute 函数可以向 MATLAB 发送命令并让 MATLAB 执行，使用此函数若需要项目名参数时，设为“EngEvalString”，否则采用默认值。

使用 DDE 的 request 函数可以向 MATLAB 请求数据，使用 poke 函数可以向 MATLAB 发送数据。

例 10.25　Visual Basic 的窗体有两个 edit 控件，控件名为 TextInput 和 TextOutput，在 TextInput 控件中输入要执行的 MATLAB 命令，然后按回车键执行，试编写 TextInput_KeyPress 方法的代码。

下面代码是 TextInput_KeyPress 方法的代码：

```
Sub TextInput_KeyPress(KeyAscii As Integer)

If KeyAscii = vbKeyReturn then
    ' 初始化 MATLAB 与 TextInput 控件的对话
    TextInput.LinkMode = vbLinkNone
    TextInput.LinkTopic = "MATLAB|Engine"
    TextInput.LinkItem = "EngEvalString"
    TextInput.LinkMode = vbLinkManual
    ' 获取 edit 控件字符串，作为 MATLAB 命令，
    szCommand = TextInput.Text
    ' 发送给 MATLAB 执行
    TextInput.LinkExecute szCommand
    TextInput.LinkMode = vbLinkNone
    ' 初始化 MATLAB 与 TextOutput 控件的对话
```

```
        TextOutput.LinkMode = vbLinkNone
        TextOutput.LinkTopic = "MATLAB|Engine"
        TextOutput.LinkItem = "EngStringResult"
        TextOutput.LinkMode = vbLinkManual
        '请求上一个 EngEvalString 命令的执行结果
        TextOutput.LinkRequest
        TextOutput.LinkMode = vbLinkNone

    End If
    End Sub
```

3．MATLAB DDE 客户端编程

当 MATLAB 作为客户端应用程序时，可以使用 MATLAB DDE 客户端函数建立和保持对话。如图 10-20 所示演示了 MATLAB 作为客户端时与服务器的通信方式。

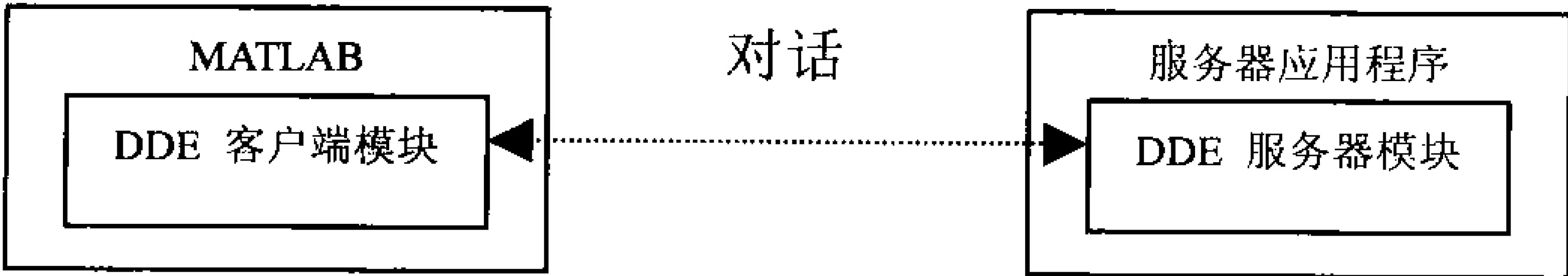


图 10-20　MATLAB 作为客户端时的通信方式

MATLAB 的 DDE 客户端模块有一系列用来通信的函数，这些函数都以 dde 为前缀，如表 10-7 所示对其进行了详细介绍。

表 10-7　MATLAB DDE 客户端函数

函数	用途
ddeadv	建立 MATLAB 与 DDE 服务器的提示链接
ddeunadv	断开 MATLAB 与 DDE 服务器的提示链接
ddeintit	初始化 MATLAB 与其他应用程序的 DDE 对话
ddeterm	终止 MATLAB 和其他应用程序的 DDE 对话
ddepoke	MATLAB 向 DDE 服务器发送数据
ddereq	MATLAB 向 DDE 服务器请求数据
ddeexec	向 DDE 服务器发送命令字符串

若服务器应用程序是 Excel 或 Word，则可以设置为 system 主题或以文件名为主题。当以文件名为主题时，Excel 文件要以.XLS 或.XLC 结尾，Word 文件以.DOC or .DOT 结尾，而且必要时包含路径。

例 10.26　创建 M 文件，实现与 Excel 建立 DDE 对话，然后向 Excel 发送一个 20×20 的矩阵。

创建 M 文件，名为 dde_excel.m，内容如下：

dde_excel.m

```
% 初始化与 Excel 的对话
chan = ddeinit('excel', 'Sheet1');
% 为 peaks 绘图创建表面
h = surf(peaks(20));
```

```
% 获取表面的 z 值
z = get(h, 'zdata');
% 设置 Excel 放置元素的范围，r1c1:r20c20 表示从第 1 行第 1 列到第 20 行第 20 列
range = 'r1c1:r20c20';
% 将 z 发送到 Excel 的电子表中
rc = ddepoke(chan, range, z);
```

执行上面的 M 文件，需要先打开 Excel，执行上面的 M 文件后，MATLAB 会先启动 figure 显示 h = surf(peaks(20));语句的结果，然后将获取的数据发送到 Excel 相应的位置。

例 10.27 从 Excel 电子表导入数据 MATLAB。

假设已有 Excel 文件 stocks.xls，这个电子表的第 3 行的 1～3 列为 3 支股票的价格，第 6～8 行的第 2 列为这 3 只股票的股数。

（1）使用下面命令初始化 MATLAB 与 Excel 的对话。

```
channel = ddeinit('excel','stocks.xls')
```

Excel 电子表的数据位置用 rxcy 格式来表示，x 为行值，y 为列值，rx1cy1:rx2cy2 表示从第 x1 行 y1 列到第 x2 行，第 y2 列的这一片矩形区域。因此，股票价格在 r3c1:r3c3 中，股数在 r6c2:r8c2 中。

（2）向 Excel 请求数据。

```
prices = ddereq(channel,'r3c1:r3c3')
prices =
          42.5015.0078.88
```

（3）然后，请求每只股票的股数。

```
shares = ddereq(channel, 'r6c2:r8c2')
shares =
               100.00
               500.00
               300.00
```

10.6 MATLAB 编译器

MATLAB 编译器可以将 MATLAB 程序转换成独立的应用程序或库并生成打包文件，使得这些应用程序和库可以在没有安装 MATLAB 软件的机器上运行。MATLAB 编译器可以编译 M 文件、MEX 文件以及其他的 MATLAB 代码，支持 MATLAB 所有的特性，包括对象、私有函数和方法。

本节介绍 MATLAB 编译器的系统配置、命令行方式和图形用户界面方式，并以一个魔方例子来介绍其具体使用。

10.6.1 MATLAB 编译器概述

本节简要介绍 MATLAB 编译器的功能和一些概念。

1．MATLAB 编译器功能

MATLAB R2007a 的编译器版本升级到了 4.6，在修补以前版本缺点的同时也增加了一

些新特性，使 MATLAB 编译器功能更加强大。使用 MATLAB 编译器可以生成以下两种。

- 可以在 UNIX、Windows 和 Macintosh 平台上可独立执行的 C/C++ 应用程序。
- C/C++共享库（Windows 环境下为动态链接库，即 DLL）。

需要注意的是，并不是所有的工具箱都是可用 MATLAB 编译器编译的，另外，对于某些工具箱，只有一些特性才可以编译，相关信息可以查询帮助。

调用 mcc 命令可以调用 MATLAB 编译器，另外，可以调用 deploytool 命令来打开配置工具（Deployment Tool），配置工具是 MATLAB 编译器、MATLAB Excel 生成器、MATLAB .NET 生成器，以及 MATLAB JAVA 生成器的图形化用户界面。本章主要介绍作为 MATLAB 编译器图形用户界面的使用，配置工具的其他功能在以后的章节介绍。

2．MATLAB 编译器技术

1）MATLAB 组件运行时间

MATLAB 编译器从 4.0 版本开始使用 MATLAB 组件运行时间（MATLAB Component Runtime，MCR）。MCR 是使 M 文件能独立运行的一些共享库的集合，为 MATLAB 语言的所有特性提供了全部支持。

MCR 使用了线程锁定，使得某一个时刻只有一个线程可以访问 MCR。所以，MATLAB 编译器生成的库文件、COM 对象和 .NET 对象对 MCR 的调用是线程安全的。

2）组件技术文件

MATLAB 编译器 4.0 也使用组件技术文件（Component Technology File，CTF）档案文件来收集配置包。在 CTF 文档中，所有的 M 文件使用高级加密标准（Advanced Encryption Standard，AES）加密。MATLAB 编译器生成的每个应用程序或共享库都有与之搭配的 CTF 档案文件，档案文件含有此应用程序或共享库库所有的 MATLAB 基础文件（M 文件和 MEX 文件）。当 CTF 档案文件解压缩到用户机器上时，仍为加密的。

3）生成过程

MATLAB 编译器创建一个软件组件的过程全部都是自动的。如创建一个独立运行的应用程序，只需提供生成此程序要用到的 M 文件清单，MATLAB 编译器就会执行以下操作。

第一步：依赖性分析（Dependency analysis）。这一步要决定作为源文件的 M 文件、MEX 文件以及 P 码文件所依赖的函数，函数类型包括源文件调用的 M 文件，以及这些 M 文件调用的 M 文件，以此类推。同时也包括 MATLAB 内嵌函数和 MATLAB 对象。

第二步：生成代码。这一步生成用来创建目标组件的所有文件代码，包括所有 M 函数的 C/C++接口代码和组件数据文件。

第三步：生成档案文件。这一步使用在依赖项分析阶段生成的 MATLAB 文件（M 文件和 MEX 文件）清单创建 CTF 档案文件。此 CTF 档案文件包含了组件执行时所需的所有文件及其路径。在配置过程中 CTF 档案文件被加密，然后压缩为单一文件。

第四步：编译。这一步将第二步中生成的 C/C++文件编译成目标文件，另外用户编写的 C/C++代码也在这一步编译。

第五步：链接。最后将第四步生成的目标文件与必要的 MATLAB 库链接，以创建最终的组件。

10.6.2 安装与配置

使用 MATLAB 编译器需要系统安装有 MATLAB 所支持的第三方 ANSI C 或 C++编译器，一般采用 Visual C++或 Borland C++。可以用命令“mbuild –setup”来简化安装过程，并可以选择一个默认的 C 编译器。此过程与 10.2.2 MEX 文件系统设置一节中编译器的设置类似，具体设置过程参见 10.2 节。

10.6.3 命令行方式

在命令行中使用 mcc 命令可以调用 MATLAB 编译器完成编译的全过程。下面介绍其用法。

（1）由 M 文件 mymfunction.m 创建独立运行的应用程序。

```
mcc -m mymfunction.m
```

此命令成功执行后，若是 Windows 环境，则会创建名为 myfunction.exe 的可独立运行的应用程序，若是其他环境，则生成的应用程序名为 myfunction。

（2）由 M 文件 mymfunction.m 创建共享库。

```
mcc -l mymfunction.m
```

此命令成功执行后，会创建一个名为 mymfunction 的共享库。若是 Windows 环境，后缀为.dll，Linux 和 Solaris 下为.so，Mac OS X 下为.dylib。

（3）由 M 文件 file1.m、file2.m 和 file3.m 创建 C 共享库。

```
mcc -l file1.m file2.m file3.m
```

成功执行后，创建库文件名为 file1，后缀与环境的关系和上一步相同。

（4）由 M 文件 file1.m、file2.m 和 file3.m 创建 C++共享库。

```
mcc -l file1.m file2.m file3.m -W cpplib -T link:lib
```

成功执行后，创建共享库文件情况和上一步相同。

10.6.4 图形用户界面方式

MATLAB 的配置工具可以以图形用户界面的形式使用 MATLAB 编译器。在控制窗口输入 deploytool 命令即可在 MATLAB 的桌面打开配置工具。使用配置工具编译文件需要 4 个步骤：创建工程、为过程添加文件、生成目标文件和打包。

例 10.28 演示用配置工具创建一个独立运行的应用程序的步骤。

输入 deploytool 命令打开配置工具，此时 MATLAB 菜单多了一个“Project”项，但是子菜单都不可用，因为还没有创建工程。创建独立运行应用程序步骤如下。

（1）新建工程

单击工具栏的🗋图标，打开新建工程对话框，界面如图 10-21 所示。

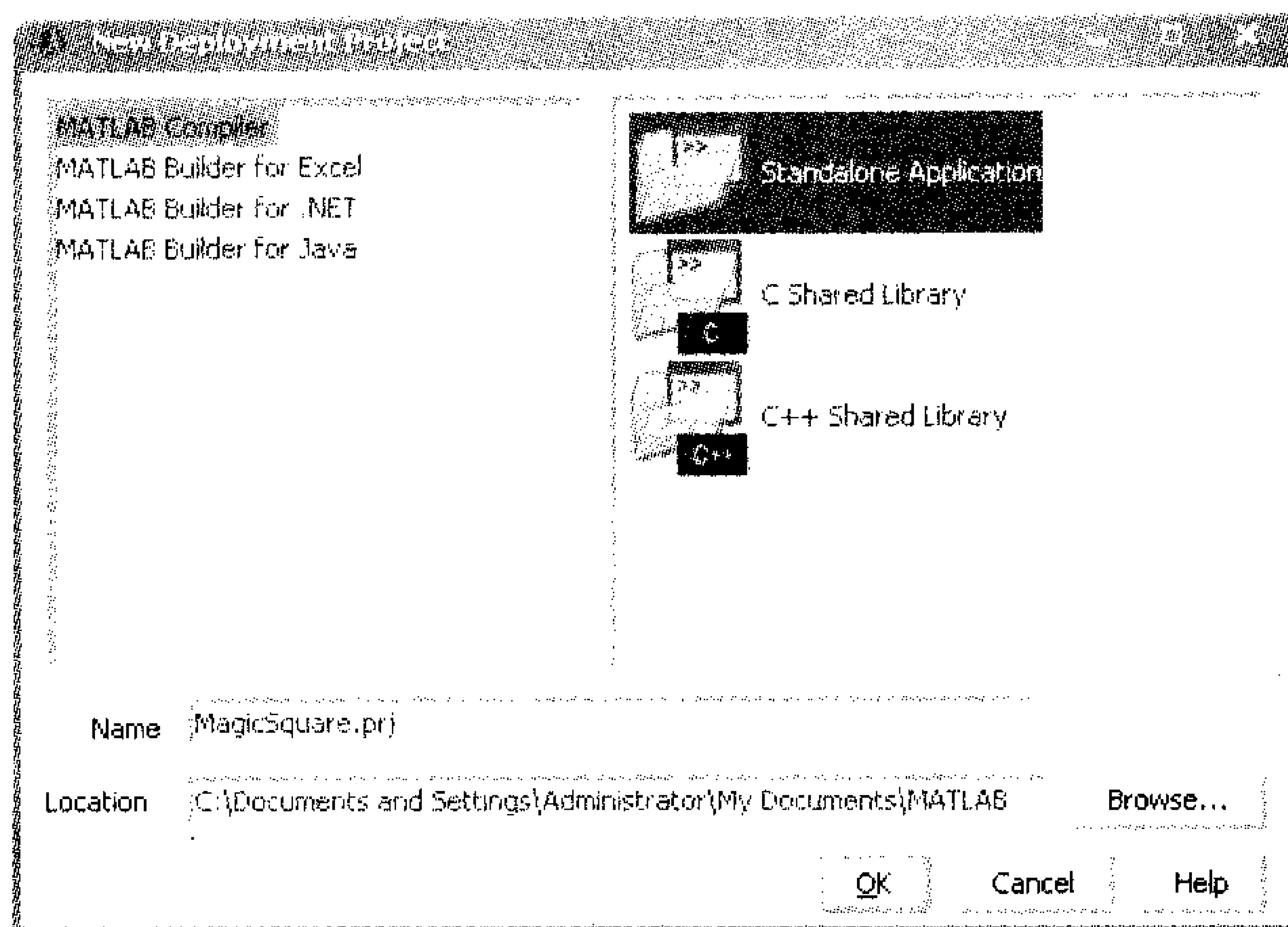


图 10-21　配置工具新建工程界面

可见配置工具是 MATLAB Compiler、Builder for Excel、Builder for .NET 和 Builder for Java 的集成开发环境，是四者共用的图形用户界面。MATLAB 编译器下有 3 个类型的工程可以创建：独立运行的应用程序、C 共享库和 C++共享库。此例选择"Standalone Application"，工程命名为 MagicSquare，注意命名不要与 MATLAB 的内嵌函数同名，如不能命名为 magic（magic 为 MATLAB 的内嵌函数），不然可能会不能正常编译。

（2）添加文件

MATLAB 编译器的配置工程的文件有 3 种：主函数（Main Function）、其他文件（Other Files）和 C/C++文件，其中主函数文件只能有一个。此工程就一个源文件：MagicSquare.m，其内容如下：

MagicSquare.m

```
function m = magicsquare(n)

if (nargin == 0)
    m = magic(5)
elseif (nargin == 1)
    m = magic(n)
end
```

此函数功能为生成并输出一个 n 阶魔方方阵，n 的值由参数确定，默认为 5。

用鼠标右键单击【Main Function】→【Add file】添加 MagicSquare.m 为主函数文件。或将此文件拖到工程中，工程会自动添加此文件为"Main Function"。另外，MATLAB 主菜单中的 Project 菜单中也有"add file"选项可以用来添加文件。

（3）创建应用程序

单击配置工具窗口工具栏中的图标，启动 build 程序，这时 MATLAB 会显示"Deployment tool Output"窗口。创建应用程序中的信息会在此窗口中显示。

最后在窗口中显示

```
Compilation completed succesfully. The output is located in D:\MyMATLAB\MagicSquare\distrib
```

说明编译成功完成。

编译成功后，在工程目录下生成 distrib 和 src 两个文件夹。其中 src 文件夹为编译此程序用到的资源以及编译时的信息等文件。而 distrib 目录下为生成的可独立运行的应用程序 MagicSquare.exe 和保证此程序可独立运行的 CTF 文档 MagicSquare.ctf。此应用程序结合其 CTF 文档就可以在无 MATLAB 环境的机器中独立运行，因为如前面所讲，CTF 文档包含了所有支持程序可以独立运行的文件。

打开 Windows 的 cmd 命令，进入到 distrib 目录下，在命令行中输入：

```
MagicSquare 5
```

运行结果如图 10-22 所示，从提示信息中可以看到，执行此文件前会先解压缩 CTF 档案文件，因为应用程序必须有 CTF 档案文件才可以独立运行。生成的应用程序成功调用了 MATLAB 的 magic 函数，生成了 5 阶魔方矩阵。

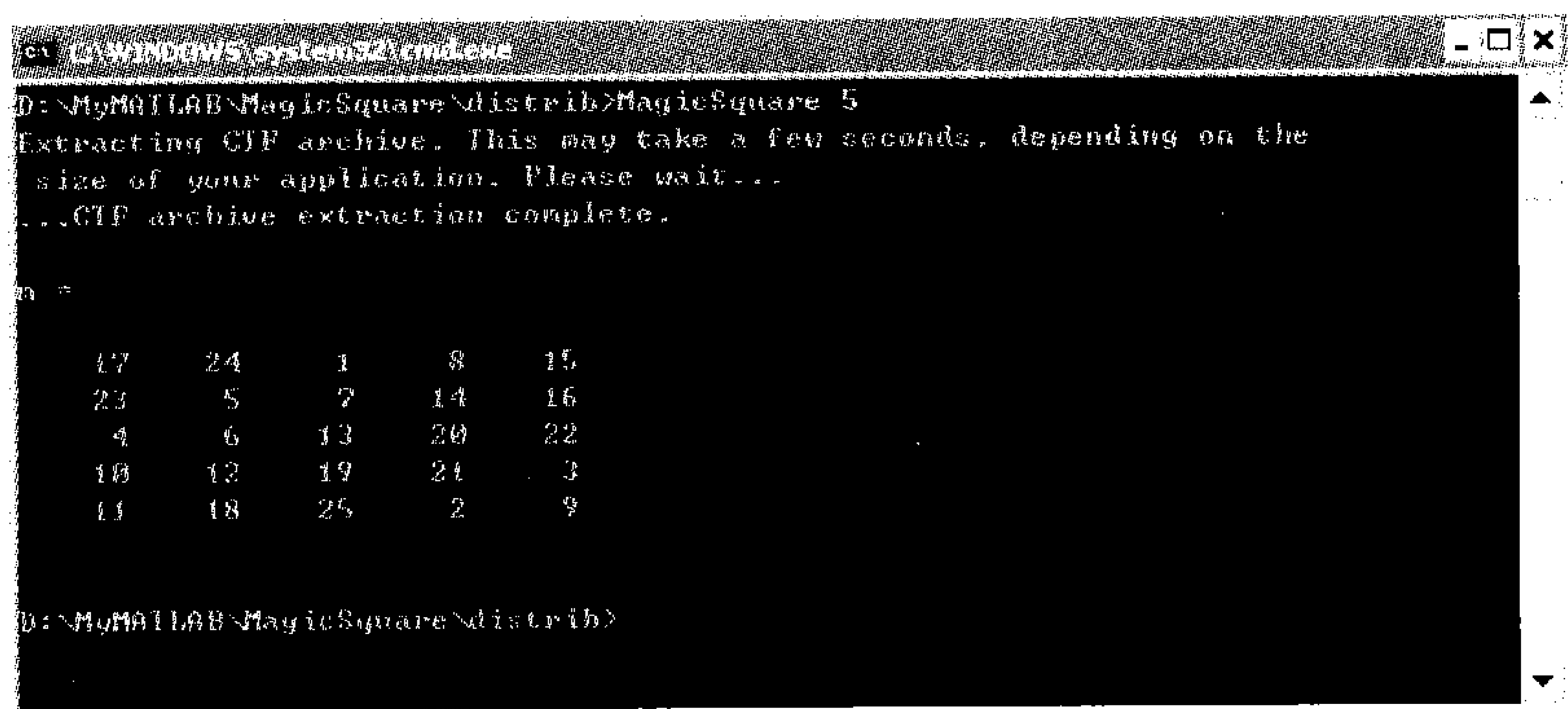


图 10-22 MagicSquare 执行结果

系统除了解压缩 CTF 档案文件之外，还在当前目录生成了 MagicSquare_mcr 文件夹，根据上面所介绍，MCR 为使文件独立运行的一些共享库的集合，为 MATLAB 语言的所有特性提供了全部支持。

（4）应用程序打包

配置工具还提供了打包功能，可以将可执行文件和独立运行所需 CTF 档案文件与库文件打包成一个单独文件，便于程序的发布。

单击配置工具栏中的图标，启动打包程序。打包程序会将 _install.bat、MagicSquare.exe 和 MagicSquare.ctf 3 个文件打包成一个文件。打包过程中会出现如图 10-23 所示界面，则打包正在进行中。

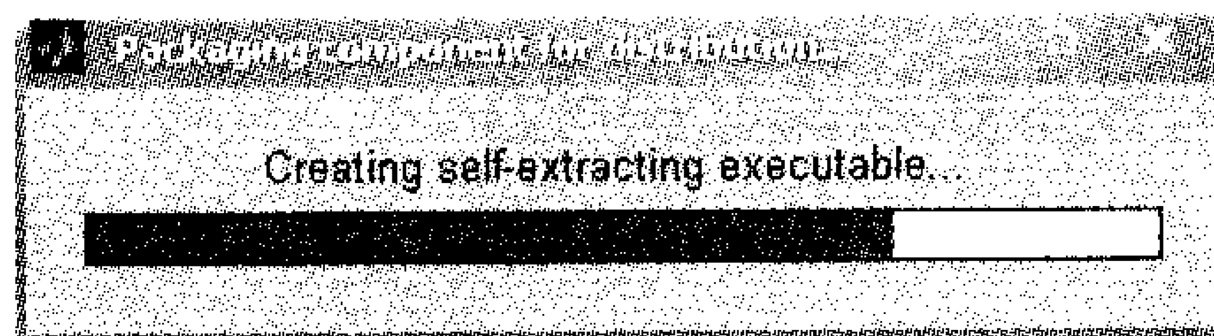


图 10-23 应用程序打包界面

打包结束后，同样是在 distrib 目录下生成 MagicSquare_pkg.exe 文件。打包文件在 Windows 环境下是一个可执行的自解压文件，运行此文件会自动安装应用程序，包括安装必要的库和自动生成 MagicSquare.exe 和 MagicSquare.ctf 文件。

10.6.5 应用实例

本节用一些实例来详细介绍 MABLAB 编译器的命令行方式和图形用户界面方式如何创建独立应用程序和 C/C++库。

1．M 文件创建独立应用程序

上面建立魔方方阵的例子就是最简单的由单独的 M 文件创建可独立运行的应用程序的情况。若只有一个 M 文件，则把此文件放在 Main Function 下即可创建应用程序。另外，对于这种情况，如例 10.28，用下面的命令也可以完成除打包外的功能。

```
mcc –mv MagicSquare.m
```

执行完毕后，也会生成可独立运行的应用程序 MagicSquare.exe（Windows 环境）。

例 10.29 由 M 文件 mrank.m 和 main.m 创建独立运行的应用程序。

M 文件内容分别如下：

mrank.m

```
function r = mrank(n)
r = zeros(n,1);
for k = 1:n
    r(k) = rank(magic(k));
end
```

main.m

```
function main
r = mrank(5)
```

用配置工具创建过程时，需将主文件 main.m 放在 Main function（Main function 下只允许放置一个文件）下，其他的 M 文件都放在 other files 下（other files 可以放置多个文件）。其他步骤与魔方方阵例子相同。

若用 mcc 命令，在控制窗口输入以下内容：

```
mcc -m main mrank
```

命令执行完毕后，会在当前目录下生成可执行文件 main.exe 和 CTF 档案文件 main.ctf，C 源代码文件 main_main.c、main_mcc_component_data.c，以及工程文件 main.prj。其中 main.prj 可以用配置工具打开编辑。

运行生成的 main.exe 文件，得到结果与原 M 文件运行结果相同。

2．M 文件混合 C/C++文件创建独立应用程序

例 10.30 M 文件和 C 文件创建独立运行运行程序，其中 C 文件调用了 M 文件中的函数。

本例演示 C 语言调用编译后 M 文件的方法。源文件在 matlabroot/extern/examples/compiler/中， M 文件有 printmatrix.m 和 mrank.m，C 文件有 mrankp.c、main_for_lib.h 和 main_for_lib.c。

将上述文件编译成一个独立运行程序，输入下面命令：

```
mcc -W lib:libPkg -T link:exe mrank printmatrix mrankp.c main_for_lib.c
```

成功执行后，会得到独立运行程序和 CTF 档案文件以及 C 源码文件。可执行文件

mrank.exe 的运行结果和例 10.29 中的 main.exe 文件相同。但是两个工程的原理不同，例 10.29 是主 M 文件调用副 M 文件，此例为 C 文件调用 M 文件完成创建魔方矩阵和显示 M 文件的返回结果等功能。

其中 C 文件 mrankp.c 内容如下：

mrankp.c

```
#include <stdio.h>
#include <math.h>
#include "libPkg.h"

main( int argc, char **argv )
{
      mxArray *N;              /* n 阶方阵 */
      mxArray *R = NULL;   /*  结果矩阵 */
      int          n;              /*   命令行的输入参数值*/

      /* 获取命令行中的输入参数，若无参数，默认为 5 */
      if (argc >= 2) {
            n = atoi(argv[1]);
      } else {
            n = 5;
      }
      /* 初始化 */
      mclInitializeApplication(NULL,0);
      libPkgInitialize();
      N = mxCreateDoubleScalar(n);

      /* 调用 mlfMrank，mlfMrank 为 mrank.m 文件编译后的文件   */
      mlfMrank(1, &R, N);
      /* 显示结果，mlfPrintmatrix.m 文件编译后的文件 */
      mlfPrintmatrix(R);

      /* 释放内存等清理工作 */
      mxDestroyArray(N);
      mxDestroyArray(R);
      libPkgTerminate();
      mclTerminateApplication();
}
```

可见此主文件分 3 个部分，初始化、调用编译后的 M 文件完成实际功能和最后释放内存等清理工作。

在生成的头文件中，mlfMrank 函数的定义为：

```
void mlfMrank(int nargout, mxArray** r, mxArray* n);
```

此函数要求有一个输入参数和一个输出参数，类型都是指针，指向 mxArray 型数据。所以函数调用：

```
R = mlfMrank(1,&R,N);
```

以 N 为输入参数，R 为输出参数，结果赋值给 R。另外，本例可以用配置工具来创建独立运行应用程序。

3．创建 C 共享库

MATLAB 编译器可以用 MATLAB 编写的算法生成 C/C++共享库（Windows 下为 DLL 文件）。从而在 C/C++ 程序中就可以以共享库的形式调用这些 MATLAB 函数，这种调用方式与在 MATLAB 控制窗口以命令行方式调用 MATLAB 函数一样方便。

例 10.31 用 MATLAB 的 M 文件实现矩阵运算，并生成相应 C 共享库，然后在 C 程序中调用此共享库完成矩阵的运算。

实现矩阵运算的 M 文件有 addmatrix.m（矩阵加法）、multiplymatrix.m（矩阵乘法）和 eigmatrix.m（求矩阵特征值），这 3 个文件都在 matlabroot/extern/examples/compiler/目录下。同目录下的 C 文件 matrixdriver.c 调用上述 M 文件生成的共享库完成矩阵运算。

1）由 M 文件生成 C 共享库

在 MATLAB 控制窗口中输入下面命令：

```
mcc -B csharedlib:libmatrix addmatrix.m multiplymatrix.m eigmatrix.m -v
```

其中-B csharedlib 选项是一个束（bundle）选项，展开后为：

```
-W lib:<libname> -T link:lib
```

其中，-W lib:<libname> 选项让 MATLAB 编译器为共享库生成函数打包文件，命名为 libname，-T link:lib 选项表示目标输出为共享库。所以上述命令以 addmatrix.m、multiplymatrix.m 和 eigmatrix.m 3 个 M 文件生成名为 libmatix 的共享库。

命令成功执行后，在当前目录下生成 C 文件 libmatrix.c 和 libmatrix.h（调用库的 C 程序要包含此文件），库文件 libmatrix.dll、libmatrix.lib 和 CTF 档案文件 libmatrix.ctf 等文件。

2）编辑 C 程序

生成 C 共享库之后，就可以在 C 程序中加以调用了。但是调用由 M 文件编译的 C 共享库有其独特的格式。调用由 MATLAB 编译器生成的共享库一般都有以下几种结构。

- 包含库文件头文件。
- 声明变量，处理输入参数。
- 调用 mclInitializeApplication 初始化并测试是否成功执行。此函数设置 MCR 全局状态，并允许创建 MCR 实例。
- 每个库调用一次<libraryname>Initialize 函数，以创建共享库所需的 MCR 实例。
- 调用共享库的函数完成实际功能，并处理返回结果。这一块是程序的主体。
- 每个库调用一次<libraryname>函数，以销毁 MCR 实例。
- 调用 mclTerminateApplication 释放全局状态 MCR 的资源 。
- 清理变量和关闭文件等，然后退出。

此例中 matrixdriver.c 调用共享库中的函数完成矩阵计算，下面分析 matrixdriver.c 文件的结构，说明如何编写调用由 MATLAB 编译器生成的共享库的 C 程序。C 文件 matrixdriver.c 内容如下（为突出文件结构，内容作了删减）：

matrixdriver.c

```
#include <stdio.h>
/* 第一个步骤 包含生成的共享库的头文件
* libmatrix.h 是由 MATLAB 编译器生成的库 libmatrix.dll 的头文件 */
#include "libmatrix.h"
```

```
void display(const mxArray* in); /* 显示结果函数 */

void *run_main(void *x)
{
    /*第二个步骤 声明变量 */
    int *err = x;
    mxArray *in1, *in2;        /* 输入参数 */
    mxArray *out = NULL;   /* 输出参数 传递给共享库的函数*/
    double data[] = {1,2,3,4,5,6,7,8,9};

    /* 第三个步骤：调用 mclInitializeApplication 函数，这一模块是固定的，可以作为模板 */
    if( !mclInitializeApplication(NULL,0) )
    {
        fprintf(stderr, "Could not initialize the application.\n");
    *err = -1;
        return(x);
    }

    /* 创建输入矩阵 代码略*/

    /* 第四个步骤 调用<libraryname>Initialize 函数，此文件 libraryname 具体化为 libmatrix. */
    if (!libmatrixInitialize()){
        fprintf(stderr,"Could not initialize the library.\n");
        *err = -2;
    }
    else
    {
        /* 第五个步骤，程序的主体，调用库中的函数来完成实际功能
        *下面分别调用 mlfAddmatrix，mlfMultiplymatrix 和 mlfeigmatrix 并显示调用结果 */
        mlfAddmatrix(1, &out, in1, in2);
        printf("The value of added matrix is:\n");
        display(out);                        /* 显示调用库函数后的结果 */
        mxDestroyArray(out); out=0;   /* 清理变量. */
        mlfMultiplymatrix(1, &out, in1, in2);
        printf("The value of the multiplied matrix is:\n");
        display(out);                        /* 显示调用库函数后的结果 */
        mxDestroyArray(out); out=0;   /* 清理变量. */
        mlfEigmatrix(1, &out, in1);
        printf("The eigenvalues of the first matrix are:\n");
        display(out);                        /* 显示调用库函数后的结果 */
        mxDestroyArray(out); out=0;    /* 清理变量. */

        /* 第六个步骤，调用<libraryname>Termination 函数 */
        libmatrixTerminate();

        /* 释放内存 */
        mxDestroyArray(in1); in1=0;
        mxDestroyArray(in2); in2 = 0;
    }
```

```
    /* 第七个步骤，调用 mclTerminateApplication */
     mclTerminateApplication();
    /* 第八个步骤，清理并退出*/
     return 0;
}

/* 显示函数 display 的具体实现，详细内容略*/
void display(const mxArray* in)
{ }
/* main 函数，具体内容略*/
int main()
{ }
```

3 个 M 文件名为 addmatrix.m、multiplymatrix.m 和 eigmatrix.m，而 C 文件 matrixdriver.c 调用 M 文件的函数语句如下：

```
mlfAddmatrix(1, &out, in1, in2);
mlfMultiplymatrix(1, &out, in1, in2);
mlfEigmatrix(1, &out, in1);
```

我们发现函数名前都加了 mlf 前缀，而且参数个数与类型都与 M 文件中的定义不同，这是因为 MATLAB 编译器生成的共享库的函数接口有其独特的命名方式。一般来说，要在 M 函数名前加前缀 mlf，参数情况与 M 文件的返回值的个数有关。

（1）M 函数无返回值。

```
void mlf<M 函数名>(<输入参数列表>);
```

（2）M 函数至少有一个返回值。

```
void mlf<M 函数名>(int 返回值个数,
                 <返回值指针列表>,
                 <输入参数列表>);
```

对于本例，addmatrix 函数生成共享库中的函数接口形式如下：

```
void mlfAddmatrix(int nlhs,mxArray **a,mxArray *a1,mxArray *a2);
```

而 M 文件中 addmatrix 函数要求输入两个矩阵值（in1 和 in2），返回一个输入矩阵和矩阵（以变量 out 存储），注意对应的 mlf 函数的第二个参数要求为指针形式，所以本例中调用 mlf 函数的方式如下：

```
mlfAddmatrix(1, &out, in1, in2);
```

3）编译 C 程序

编写好 C 程序后，需要编译成可执行程序，在 MATLAB 控制窗口中输入下面命令：

```
mbuild matrixdriver.c libmatrix.lib      (Windows 环境)
```

编译成功后，会生成 libmatix.exe 可执行文件。

4）测试程序

编译后得到可执行文件后，在 MATLAB 控制窗口输入下面命令执行程序：

```
!libmatix
```

首先会显示解压缩 CTF 文件，接着显示文件执行结果如下：

```
The value of added matrix is:
2.00  8.00  14.00
4.00  10.0016.00
6.00  12.0018.00
```

```
The value of the multiplied matrix is:
30.0066.00102.00
36.0081.00126.00
42.0096.00150.00

The eigenvalues of the first matrix are:
16.12-1.12 -0.00
```

4．创建 C++共享库

例 10.32　用 C++语言编写例 10-31。

此例与例 10.31 不同的是，这次用 C++语言编写调用共享库的程序。生成 C++共享库和生成 C 共享库的步骤类似，将例 10.31 中 3 个 M 文件生成 C++共享库的命令如下：

```
mcc -W cpplib:libmatrixp -T link:lib addmatrix.m multiplymatrix.m eigmatrix.m -v
```

其中，-W cpplib:<libname> 选项表示 MATLAB 编译器生成一个名为 libname 的 C++共享库，-T link:lib 选项表示生成的目标文件为共享库。

命令成功执行后，生成文件与上例类似，唯一不同的是上例生成的共享库源文件为 C 文件，此例生成的是 CPP 文件。

同样，本例的 matrixdriver 程序用的是 C++版本，大体结构与 C 版本一样。C++的版本中数组用类 mwArray 的对象表示，每个 mwArray 对象包含一个 MATLAB 数组结构体的指针。因此，mwArray 对象的属性是 MATLAB 数组属性的一个父集。

下面列出 matrixdriver.cpp 的关键代码，分析 C++版本与 C 版本程序的异同。

matrixdriver.cpp

```
// 第一个步骤 包含共享库的头文件
#include "libmatrixp.h"

void *run_main(void *x)
{
    // 第二步骤 创建变量，检查输入参数
    int *err = (int *)x;
    if (err == NULL) return 0;

    //第三个步骤：调用 mclInitializeApplication 函数，这一模块是固定的，可以作为模板
    if (!mclInitializeApplication(NULL,0))
    {
        std::cerr << "could not initialize application properly"
                  << std::endl;
     *err = -1;
        return x;
    }
    //第四个步骤 调用<libname>Initialize 进行初始化
    if( !libmatrixpInitialize() )
    {
        std::cerr << "could not initialize library properly"
                  << std::endl;
     *err = -1;
    }
```

```
        else
        {
            try
            {
                // 创建输入数据
                double data[] = {1,2,3,4,5,6,7,8,9};
                mwArray in1(3, 3, mxDOUBLE_CLASS, mxREAL);
                mwArray in2(3, 3, mxDOUBLE_CLASS, mxREAL);
                in1.SetData(data, 9);
                in2.SetData(data, 9);

                // 创建输出数组
                mwArray out;

                // 第五步骤 调用库中函数完成实际功能
                // 下面分别调用 addmatrix，multiplymatrix 和 eigmatrix 并显示调用结果

                addmatrix(1, out, in1, in2);
                std::cout << "Value of added matrix is:" << std::endl;
                std::cout << out << std::endl;

                multiplymatrix(1, out, in1, in2);
                std::cout << "The value of the multiplied matrix is:"<< std::endl;
                std::cout << out << std::endl;

                eigmatrix(1, out, in1);
                std::cout << "The eigenvalues of the first matrix are:"<< std::endl;
                std::cout << out << std::endl;
            }
            catch (const mwException& e)
            {
              std::cerr << e.what() << std::endl;
              *err = -2;
            }
            catch (...)
            {
              std::cerr << "Unexpected error thrown" << std::endl;
              *err = -3;
            }
            // 第六步骤 调用<libname>Termination 函数
            libmatrixpTerminate();
        }
        //第七个步骤 调用 mclTerminateApplication 函数
        mclTerminateApplication();
        //第八个步骤，清理，退出程序
        return 0;
}

int main() //main 代码略
{}
```

由上面的 C++版本可以看出与 C 版本的不同有以下几个方面。

（1）C++文件与 MATLAB 的接口数据类型为 mwArray 而不是 C 文件的 mxArray 类型。

（2）C++版本的实际操作都在 try 模块中完成。这是因为任何异常都应由函数调用者处理。

（3）C++共享库与 C 共享库中的接口函数命名方式不同。C++共享库为每个 M 函数生成两个接口函数集合，一个是 C 版本，一个是 C++版本。C 版本与上例中生成的 C 共享库相同（以 mlf 以前缀），而 C++版本的接口函数的命名方式同样和对应 M 函数的返回值个数有关：

①M 函数无返回值。

```
void <function-name>(<输入参数列表>);
```

②M 函数至少一个返回值。

```
void <function-name>(int 返回值个数,
                     <返回值列表>,
                     <输入值列表>);
```

这种情况下，<输入参数列表>表示由逗号分割的类型为 const mwArray&的变量表，而<返回值列表>的变量为 mwArray&类型。

从上面可以看出，C++版本的函数命名方式中函数名和函数参数都与 C 版本有很大不同，首先函数名没有了 mlf 前缀，再者当函数有返回值时，C++的返回值的列表不是 C 版本中的指针类型。

如在 libmatrix 库中，addmatrix 的函数 C++接口为：

```
void addmatrix(int nargout, mwArray& a , const mwArray& a1, const mwArray& a2);
```

所以 matrixdriver.cpp 中对 addmatrix 函数的调用方法为：

```
addmatrix(1, out, in1, in2);
```

读者可以对照 C 代码和 C++代码以发现其中的异同。

编译 matrixdriver.cpp C++程序，必须使用 C++编译器。编译命令与 C 版本的一样：

```
mbuild matrixdriver.cpp libmatrixp.lib              (Windows 环境)
```

C++版本的程序需要处理调用 C++共享库的异常，为了能够正确处理，最好使用 C++ try-catch 模块，将实际代码放入 try 子模块中，catch 模块使用 mwException 类获取异常。典型的 try-catch 模块如下（标准模式，可作为模板）：

```
try
{
     (调用 C 共享库中函数)
}
catch (const mwException& e)
{
     (处理错误和异常)
}
```

C++文件 matrixdriver.cpp 演示了调用 C++共享库中函数时典型的处理异常方法。读者可以仿照此例来编写自己的代码。

10.7 MATLAB 硬件接口技术

MATLAB 提供了访问硬件的接口函数与方法，使得 MATLAB 可以和计算机外设方便地进行通信，大大扩展了 MATLAB 的应用领域。MATLAB 与硬件接口包括串口通信和并口通信两种。

本章主要介绍 MATLAB 的串口通信，结合实例介绍了 MATLAB 串口通信编程的步骤和方法。

10.7.1 MATLAB 串口接口概述

MATLAB 串口接口使得 MATLAB 可以直接访问外部设备，如调制解调器、打印机和科学计算仪器等可以连接到计算机串口上的设备。串口接口由串口对象来建立，此对象支持以下功能和属性。

- 配置串口通信。
- 使用串口控制栓（Control Pins）。
- 读/写数据。
- 使用事件和回调函数。
- 将信息存储到硬盘上。

利用 Instrument Control 工具箱，可以使用串口的其他功能。Instrument Control 工具箱提供了图形化工具 Test & Measurement，使得用户不编写代码就可以与外设通信，Test & Measurement 工具可生成用重用的 MATLAB 代码。

MATLAB 串口对象支持 RS-232、RS-422 和 RS-485 等串口接口标准，其中最常用的标准是 RS-232。下面的讲解都使用 RS-232 标准。MATLAB 串口支持 Windows、Linux 和 Sun Solaris 等平台。

编程前，需要查询本机上串口的信息。在 Windows 环境下打开控制面板，依次选择“系统”、“硬件”、“设备管理器”，在打开的设备管理器中，展开“端口”结点，双击“通讯端口（COM1）”即可打开本节串口的信息对话框，用户可以对其进行查询和属性设置，如图 10-24 所示。

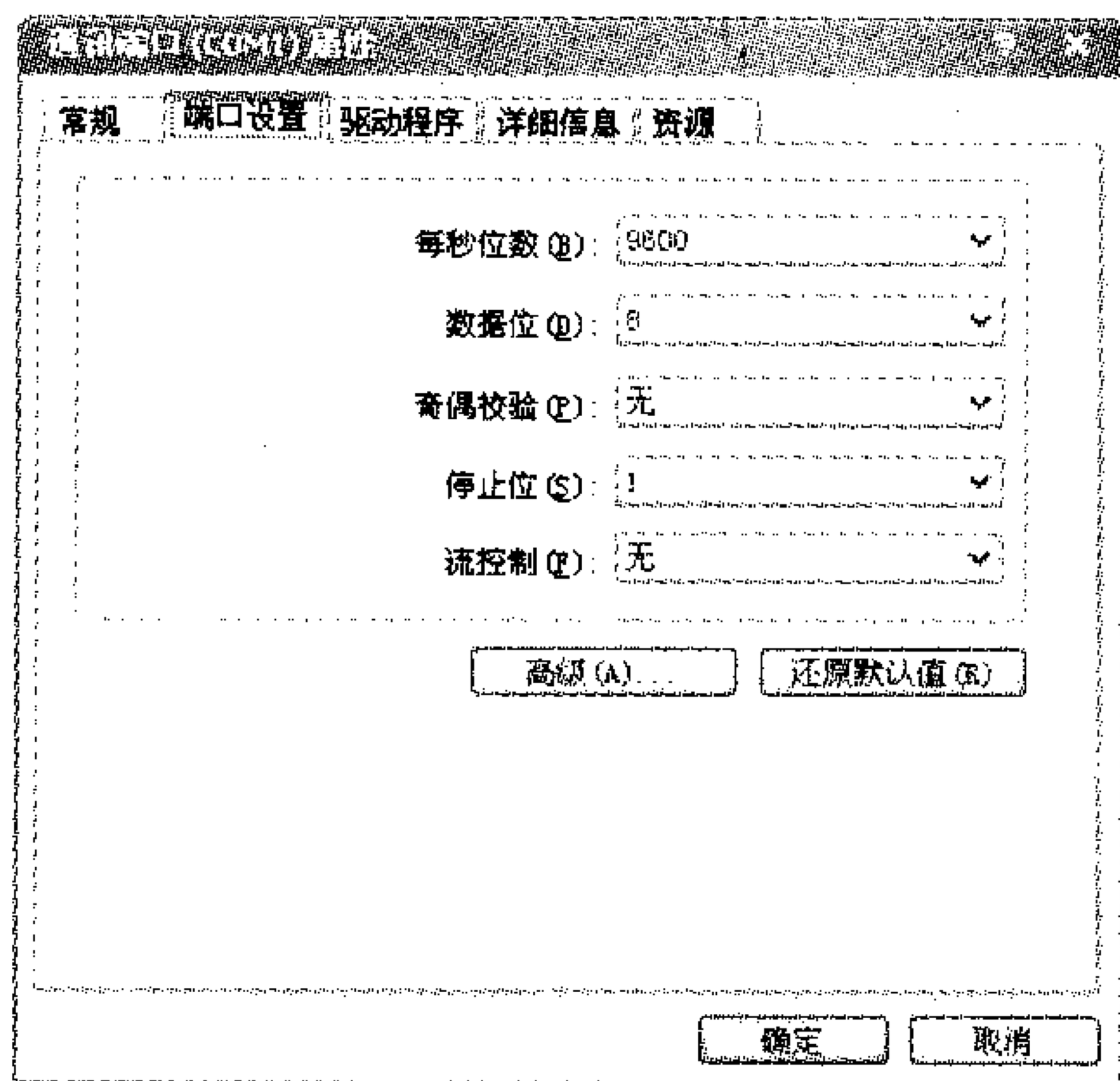


图 10-24 Windows 下端口属性对话框

本章不介绍串口通信的基本知识，如串口通信概念、串口通信标准、如何用电缆连接两个设备、串口通信的数据格式和串口传送方式等。读者若对串口通信的预备知识没有掌握，可以查阅相关书籍，本章主要介绍 MATLAB 串口通信编程。

10.7.2 MATLAB 串口通信编程

串口编程主要的步骤有以下几方面。

（1）创建串口对象。

（2）连接已有的设备。

（3）配置属性。

（4）读/写数据－程序的主体，完成实际功能。

（5）断开连接并清理。

下面是简单的一个串口通信程序，包含了上述 5 个步骤。

Serialexa.m

```
s = serial('COM1');          %    创建串口对象
set(s,'BaudRate',4800);      %    设置其属性值
fopen(s);                    %    连接设备
fprintf(s,'*IDN?')           %    读/写数据
out = fscanf(s);             %
fclose(s)                    %    断开连接
delete(s)                    %    清理
clear s                      %
```

下面详细介绍每个步骤。

1. 创建串口对象

用 serial 函数为要通信的串口创建对象，serial 函数创建串口对象需要指定串口名作为输入参数。比如，创建串口 COM1 的串口对象，输入下面命令：

```
s = serial('COM1');
```

另外，在创建串口对象同时，可以设置某些属性值，比如串口的波特率和数据位等属性。方法是在串口名后添加要设置的“属性名/属性值”对象，命令如下：

```
s = serial('COM1','BaudRate',4800,'Parity','even')
```

2. 获取/设置属性值

串口对象创建之后，用 set、get 和句号符号表达式可以获取/设置串口对象的属性值。下面具体介绍。

1）获取属性值

显示当前一个属性值，可以使用 get 命令加属性名，命令如下：

```
get(s,'OutputBufferSize')
ans =
         512
```

若显示当前的多个属性值，可将属性名作为细胞数组的元素，命令如下：

```
get(s,{'Parity','TransferStatus'})
ans =
```

```
'none'    'idle'
```

可以使用句号符号表达式显示单个属性值，命令如下：

```
s.Parity
ans =
    none
```

2）设置属性值

设置单个属性值，可以使用 set 函数加属性名，命令如下：

```
set(s,'BaudRate',4800);
```

或者使用句号符号表达式：

```
s.BaudRate = 4800;
```

使用句号符号表达式只能为单个属性值赋值，要设置多个属性值，需要 set 函数，将要设置的多个属性值以“属性名/属性值”对的形式排列，命令如下：

```
set(s,'DataBits',7,'Name','Test1-serial')。
```

3）指定属性名

串口的属性名不区分大小写，增加了属性名的可读性。并且一些属性有其缩写形式，只需要一部分字母就可以唯一确定此属性。下面的方法都可以设置 BaudRate 的属性值：

```
set(s,'BaudRate',4800)
set(s,'baudrate',4800)
set(s,'BAUD',4800)
```

在 M 文件中最好使用属性的全名，以免 MATLAB 升级后这些缩写形式不再支持。

4）默认属性值

若没有对属性进行设置，此属性使用默认值。所有可设置的属性值都有其默认值。

操作系统提供了如波特率等属性的默认值，但是这些属性可能被 MATLAB 所修改，而在串口应用程序中变得不可用。

若某个属性只有限个属性值可以设置，在用 set 函数查看此属性可设置的属性值时，其默认值是用{}括起来的。如 Parity 属性的默认值是 none：

```
set(s,'Parity')
[ {none} | odd | even | mark | space ]
```

3．连接/断开串口设备

对象连接之后，才能配置对象的属性值和读/写数据。当不再使用串口对象时，需要断开连接并作清理工作。

连接串口设备可以使用下面语句：

```
fopen(s)
```

若连接成功，此时查询串口对象的状态：

```
s.Status
ans =
    open
```

而断开可使用：

```
fclose(s)
```

断开成功，此时查询串口对象状态：

```
s.Status
ans =
    closed
```

断开与设备连接后，还需要一些清理工作，包括从内存中和 MATLAB 工作空间中将串口对象删除，用下面语句：

```
delete(s)  %  清除内存中串口对象
clear s    %  清除 MATLAB 工作空间串口对象
```

4．读/写数据

读/写数据是串口编程的主体，完成了与串口设备数据的交换。

1）读数据

表 10-8 和表 10-9 分别列出了与读数据有关的函数和属性。

表 10-8 读数据有关的函数

函数	用途
fgetl	读取一行文本，不包括结束符
fgets	读取一行文本，包括结束符
fread	读取二进制数据
fscanf	读取数据，然后格式化为文本
readasync	异步方式读取数据

表 10-9 读数据有关的属性

函数	用途
BytesAvailable	输入缓冲可用的字节数
InputBufferSize	输入缓冲大小
ReadAsyncMode	异步读模式，值可为 continuous 或 manual
ValuesReceived	读取数据总个数

使用 fgetl、fgets 以及 fscanf 函数读取设备的数据并转换为文本格式。默认情况下，fscanf 函数读取数据格式为“%c”，因为一般串口返回的数据都是基于文本格式的。

属性 ReadAsyncMode 取值有 continuous 和 manual。若设置为 continuous（默认值），则串口对象会连续的查询串口是否有可读数据。有数据可读时，数据存储在输入缓冲。

将输入缓冲的数据传送到 MATLAB，可以使用同步读函数，如 fgetl 和 fscanf。若有数据可读，下面的代码很快返回数据，如下面代码：

```
s.ReadAsyncMode = 'continuous';
fprintf(s,'*IDN?')
s.BytesAvailable
ans =
     56
out = fscanf(s);
```

若属性 ReadAsyncMode 设置为 manual，串口对象并不连续的查询。用 readasync 函数异步读数据，然后同步读函数将数据从输入缓冲传送到 MATLAB，如下面代码：

```
s.ReadAsyncMode = 'manual';
fprintf(s,'*IDN?')
s.BytesAvailable
ans =
```

```
        0
readasync(s)
s.BytesAvailable
ans =
      56
out = fscanf(s);
```

MATLAB 命令行调用 fscanf 函数的读操作完成的标志有以下几方面。

- 读到了结束符。
- 属性 Timeout 规定的时间到。
- 已经读完预规定的数据个数。
- 输入缓冲已满。

用 fread 函数可读取设备的二进制数据，这意味着将数值数据返回给 MATLAB。如要返回示波器的 cursor 和 display 设置，可先向设备写“CURSOR?”和“DISPLAY?”命令，然后读取此命令的返回值，即可获得 cursor 和 display 的设置值。

将数据传送给 MATLAB 可使用任意的同步读函数，若使用 fgetl、fgets 或 fscanf，必须使用两次读操作，因为输入缓冲有两个结束符。但是若使用 fread，调用一次即可传送所有数据。

fread 函数默认返回的是双精度的数值数据，可以用 MATLAB 的 char 函数将数据转换为字符串数据，如下面代码：

```
val = char(out)'
val =
      HBARS;CH1;SECONDS;-1.0E-3;1.0E-3;VOLTS;-6.56E-1;6.24E-1 YT;DOTS;0;45
```

2）写数据

向设备写数据可使用 fprintf 和 fwrite 函数，不同的是 fprintf 函数向设备写的是文本数据，fwrite 函数写的是二进制数据。如表 10-10 所示列出了与写数据有关的属性。

表 10-10　写数据有关的属性

函数	用途
BytesToOutput	向设备写文本数据
OutputBufferSize	输出缓冲的字节数
Timeout	等待读写操作完毕时间
TransferStatus	是否正在进行异步读写操作
ValuesSent	预写入数据的总个数

使用 fprintf 函数可以向设备写入文本数据，一般传送命令字符串以改变设备的设置、让设备准备返回数据和状态信息等。如下面命令改变了示波器的显示对比度:

```
fprintf(s,'Display:Contrast 45')
```

使用 fwrite 函数来写二进制数据，这里写二进制数据即为写数值数据。有些串口设备值接受基于文本的命令，所以有时需要 fprintf 函数完成所有的写操作。

当数据完全写入或 Timeout 属性规定的时间后，就视为 fprintf 和 fwrite 函数执行完毕。默认情况下，fprintf 函数写数据采用“%s\n”格式，因为某些串口设备仅仅接受基于文本的命令。查询 ValuesSent 属性可以得知传送字符的个数，如下所示：

```
s.ValuesSent
ans =
     20
```

需要注意的是，ValuesSent 属性值把结束符也计算在内，因为字符串中的每个“\n”出现都被转换成了结束符。结束符的默认值为换行符（Line feed，LF)。下面命令查询当前结束符值：

```
s.Terminator
ans =
     LF
```

默认情况下，fprintf 和 fwrite 函数以同步方式执行，即执行后 MATLAB 命令行被锁定，直到执行完毕。若以异步方式写数据，需在 fprintf 和 fwrite 函数加入参数 async，比如：

```
fprintf(s,'Display:Contrast 45','async')
```

异步方式不锁定 MATLAB 命令行。另外，在进行异步写数据的同时还可以执行异步的读操作和使用所有支持的回调函数。若没有异步操作在进行，串口对象的 TransferStatus 属性值为 idle。

10.7.3 编程实例

下面用 3 个例子介绍 MATLAB 硬件接口的使用方法。

1．读/写文本数据

例 10.33 演示通过读/写文本数据与串口设备交互。

本例中的串口设备是一个双信道的示波器 Tektronix TDS 210，连接到 COM1 端口。本例中的命令也是此设备特有的，正弦波输入到示波器的 2 信道，工作为测量此信号的电压。

（1）创建串口对象——创建串口 COM1 相关的串口对象。

```
s = serial('COM1');
```

（2）连接设备——连接到示波器。由于 ReadAsyncMode 属性的默认值为 continuous，所以设备一有可用数据就会传送到输入缓冲。

```
fopen(s)
```

（3）读写数据——用 fprintf 向设备写“*IDN?”命令，然后用 fscanf 命令读回此命令的结果。

```
fprintf(s,'*IDN?')
idn = fscanf(s)
idn =
     TEKTRONIX,TDS 210,0,CF:91.1CT FV:v1.16 TDS2CM:CMV:v1.04
```

下面确定测量源，可能的测量源包括此示波器的 1 号信道和 2 号信道。

```
fprintf(s,'MEASUREMENT:IMMED:SOURCE?')
source = fscanf(s)
source =
     CH1
```

结果由 1 号信道返回，由于信号连接到了 2 号信道，所以设置成由 2 号信道返回：

```
fprintf(s,'MEASUREMENT:IMMED:SOURCECH2')
fprintf(s,'MEASUREMENT:IMMED:SOURCE?')
source = fscanf(s)
```

```
source =
          CH2
```

下面是请求设备的电压数据：

```
fprintf(s,'MEASUREMENT:MEAS1:TYPEPK2PK')
fprintf(s,'MEASUREMENT:MEAS1:VALUE?')
```

用 fscanf 将输入缓存的数据传送给 MATLAB：

```
ptop = fscanf(s,'%g')
ptop =
        2.0199999809E0
```

（4）断开连接并清理——不再使用设备时，断开连接，从内存清除，然后从 MATLAB 工作空间删除。

```
fclose(s)
delete(s)
clear s
```

2．使用 strread 解析输入数据

例 10.34 演示使用 strread 函数解析从设备读取的数据。

若将一个字符串分成一个或多个变量，可用 strread 函数完成。本例的设备与例 10.33 相同，连接到串口 COM1。

（1）创建串口对象和连接设备与例 10.33 相同。

（2）读/写数据——写“RS232?”命令，然后读返回结果。RS232? 查询 RS-232 的设置并返回波特率、软件流控制设置、硬件流控制设置、奇偶类型和终止符。

```
fprintf(s,'RS232?')
data = fscanf(s)
data = 9600;0;0;NONE;LF
```

（3）使用 strread 函数将上述结果解析为 5 个新的变量。

```
[br,sfc,hfc,par,tm] = strread(data,'%d%d%d%s%s','delimiter',';')
br =
    9600
sfc =
    0
hfc =
     0
par =
     'NONE'
tm =
     'LF'
```

（4）最后断开连接并清理，与例 10.33 相同。

3．读取二进制数据

例 10.35 演示如何将 TDS 210 示波器的显示下载到 MATLAB 中。

显示数据以 Windows 的 bitmap 格式传送并存储到硬盘。此数据格式提供了永久记录方式，并且是记录重要的信号和参数的简单方法。

（1）在创建串口对象后，在连接设备之前需要设置串口设备的属性。因为要传送的数据量比较大，设置输入缓冲足够大以接受足够多字节数据，另外设置波特率为设备所支持

的最高值。

```
s.InputBufferSize = 50000;
s.BaudRate = 19200;
```

（2）读/写数据——完成串口串口对象，设置属性和连接设备后，即可读/写数据。将显示屏上的视图以 btmap 的格式传送：

```
fprintf(s,'HARDCOPY:PORT RS232')
fprintf(s,'HARDCOPY:FORMAT BMP')
fprintf(s,'HARDCOPY START')
```

当所有数据都传送到输入缓冲后，再将数据以无符号 8 位整数传送到 MATLAB 工作空间：

```
out = fread(s,s.BytesAvailable,'uint8');
```

（3）断开设备并清理。

10.8 MATLAB Web 服务

Web 服务包括基于 XML 的一套技术，可以通过网络调用远程程序。网络可以指当地局域网或万维网。简而言之，Web 服务可使的程序在不同的操作系统或平台之间实现交互。MATLAB 实现了 SOAP（Simple Object Access Protocol）和 WSDL（Web Services Description Language）两种 Web 服务技术，MATLAB 可作为 Web 服务的客户端，向服务器发送请求并处理响应。

本节主要介绍 MATLAB 用过 SOAP 和 MSDL 调用 Web 服务方法，最后通过实例讲解了创建用 Web 服务的 MATLAB 应用程序。本节内容要求读者掌握 SOAP 和 WSDL 的基础知识。

10.8.1 MATLAB Web 服务使用

MATLAB 中使用 createClassFromWsdl 函数调用 Web 服务的方法。此函数创建了基于 Web 服务 API 方法的 MATLAB 类。举例如下：

```
createClassFromWsdl('http://example.com/service.wsdl')
```

上面的例子用 URL 中的 WSDL 文件作为参数，也可以直接用本地文件路径作为参数：

```
createClassFromWsdl('\myservicedirectory\service.wsdl')
```

但是目标文件必须是 WSDL 文件。

下面逐步介绍如何创建一个简单的 Web 服务。

（1）在 Web 浏览器中，输入 http://www.xmethods.net/，进入 XMethods 网站。

（2）然后在“XMethods Demo Services”区，单击网页底部的“Currency Exchange Rate”链接。

（3）在打开的“Currency Exchange Rate”网页中，找到“WSDL URL”，单击“View RPC Profile”链接，在 RPC Profile 网页，找到可用的方法。

此时，可用的方法为 getRate。注意其输入/输出参数以及参数的数据类型。

（4）在 MATLAB 控制窗口输入下面命令，将 WSDL URL 传递给函数

createClassFromWsdl，以创建 CurrencyExchangeService 类。

```
createClassFromWsdl('http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl')
ans =
    CurrencyExchangeService
```

使用 MATLAB 的 methods 函数查看 CurrencyExchangeService 类的方法：

```
methods(CurrencyExchangeService)
```

得到下面结果：

```
Methods for class CurrencyExchangeService:
CurrencyExchangeService      getRate      display
```

在 MATLAB 的当前目录会发现生成了@CurrencyExchangeService 文件夹，文件夹下有 3 个文件。

- CurrencyExchangeService.m：MATLAB 类的构造函数。
- display.m：显示结果的方法。
- getRate.m：getRate 方法的 M 文件。

函数 createClassFromWsdl 会自动创建对应于 Web 服务每个方法的 M 文件，另外创建一个显示结果的 display.m 文件，一个 MATLAB 类的构造函数文件。使用 MATLAB 的 help 函数可查询这些文件的信息，如输入以下命令：

```
help CurrencyExchangeService/getRate
```

得到以下信息：

```
getRate(obj,country1,country2)
        Input:
          country1 = (string)
          country2 = (string)

        Output:
          Result = (float)
```

调用构造函数创建对象：

```
ces = CurrencyExchangeService;
```

调用 getRate 方法得到美国和法国之间的汇率：

```
getRate(ces, 'USA', 'France')
ans =
    5.1127
```

然后调用 getRate 方法得到阿根廷和智利之间的汇率：

```
getRate(ces, 'Argentina', 'Chile')
ans =
    172.3052
```

10.8.2 创建有 Web 服务的 MATLAB 应用程序

创建有 Web 服务的 MATLAB 应用程序，一般用 Web 服务导入数据。创建有 Web 服务的应用程序前，了解下面知识。

1．Web 服务的限制

即使在本书的写作中，Web 服务技术仍在继续发展变化中。下面列出了创建有 Web 服

务的 MATLAB 应用程序的限制。

- 大部分的 Web 服务通过 HTTP。正如 Internet 本身一样，服务质量不能绝对保证，所以创建的应用程序或许是不可靠的。
- Web 服务和像 WSDL 和 SOAP 等相关的技术都是新兴技术，所以一直在不断发展变化着。
- 若调用远程 Web 服务，在之前检验其有效性和可靠性。另外，现在免费的 Web 服务以后可能不再免费，甚至不再提供。

2. Web 服务编程

由于 Internet 本身固有的不可预见性，在 Web 服务编程时务必格外注意。降低风险的一个有效方法就是正确使用程序控制和错误处理模块，具体包括下面内容。

1）Try-Catch 语句

Try-Catch 语句可以捕获在调用方法时的错误。下面的代码演示了其用法：

```
try
    r = getRate(CurrencyExchangeService, 'USA', 'France');
catch
    r = Nan;
    disp(lasterr);
end
```

2）if 语句

if 语句可以检查表达式或语句的真值。下面例子用 if 语句获取本地的 WSDL：

```
wsdlUrl = 'http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl';
wsdlFile = 'CurrencyExchangeService.wsdl';
if ~(exist(wsdlFile,'file') == 2)
    urlwrite(wsdlUrl,wsdlFile);
end
```

3）Error 函数

Error 函数可以抛出特定的错误。下面代码段演示了 try-catch 语句中 error 函数的使用方法：

```
try
    r = getRate(CurrencyExchangeService, 'USA', 'France');
catch
    error('Could not return exchange rate');
end
```

3. 简单的 M 文件例子

例 10.36 编辑 M 文件，利用 Web 服务的方法计算两国之间的汇率，并找出最大和最小的汇率。

此例中的 M 文件 ratesam.m 是脚本文件，演示了 Web 服务编程的一般过程。

ratesam.m

```
% 以国家名创建细胞数组
C = { 'USA', 'France '; 'Argentina', 'Chile'; 'Morocco', 'Portugal'; 'Germany', 'Denmark'};
rates = {};    % 创建空数组存储返回的汇率值
```

```
wsdlUrl = 'http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl';
wsdlFile = 'CurrencyExchangeService.wsdl';
if ~(exist(wsdlFile, 'file') == 2)
    urlwrite(wsdlUrl, wsdlFile);
end

try
    createClassFromWsdl(wsdlFile);      % 从 WSDL 创建类
catch
    error('Unable to create WSDL class');
end

ces = CurrencyExchangeService;

count = length(C);
for x = 1:count
    try
        r = getRate(ces, C{x,:});    % 调用 getRate 方法
    catch
        r = NaN;
        warning(lasterr);
    end
    rates = horzcat(rates, r);
end

for x = 1:count
    disp(['Exchange rate for ' C{x,1} ' and ' C{x,2} ':'])
    disp(rates{x})
end
disp(' ')

% 显示最高汇率
disp('Highest exchange rate:');
disp(max([rates{:}]));

% 显示最低汇率
disp('Lowest exchange rate:');
disp(min([rates{:}]));
```

10.9 MATLAB 与 Excel 接口

MATLAB 作为功能强大的数学软件，数据计算和图像显示是其优势，而微软的 Excel 同样具有较强的数据统计和显示能力。Excel 在一些较为简单的图像显示中方便易用，对一些复杂的图像显示，特别是三维数据显示就差强人意了。MATLAB 提供了两种方法实现了

与 Excel 的数据共享和功能交互，使这两大数学软件有机地结合在了一起。

本章分别介绍 MATLAB 与 Excel 的两种接口方式，Excel Link 和 Excel 生成器，并结合实例介绍其用法。

10.9.1 MATLAB Excel link

1. Excel Link 概述

Excel Link 是一个插件软件，在基于 Windows 的环境中集成了 Excel 和 MATLAB。使 Excel 和 MATLAB 协作，可以在 Excel 的工作表和宏工具中使用 MATLAB 的数值分析、数学计算和绘图功能。Excel Link 保持了二者之间数据的同步交换。

利用 Excel Link 实现 Excel 工作空间与 MATLAB 工作空间通信时，不必脱离 Excel 环境，MATLAB 在后台运行。在 Excel 电子表和宏中可以使用 Excel Link 的函数，这些函数实现了 MATLAB 与 Excel 的数据交换与处理，使 Excel Link 显得简单而高效。

Excel Link 的工作原理与运行机制如图 10-25 所示。

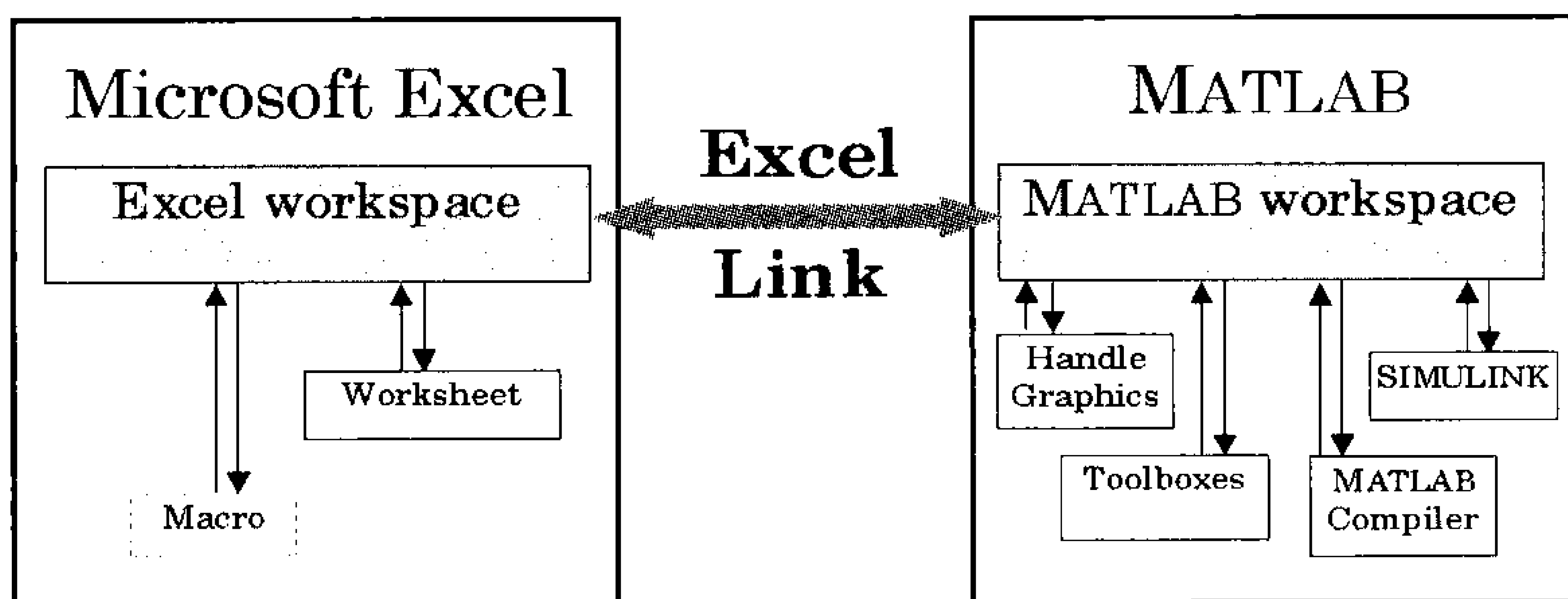


图 10-25　Excel Link 的工作原理

2. 安装与配置

Excel Link 的工作环境必须是微软的 Windows，如 Windows 98、Windows 2000 和 Windows XP，系统必须安装 Excel 和 MATLAB，而且安装 MATLAB 时选择安装 Excel Link。

上述系统要求具备后，还需对 Excel Link 进行配置，以实现 MATLAB 与 Excel 的连接。具体步骤操作如下所示，只需要设置一次，以后就可以使用。

启动 Excel，执行【工具】→【加载宏】命令，然后单击【浏览】按钮，在 matlabroot/toolbox/exlink 目录下找到并单击 excllink.xla（此时窗口会显示 Excel Link 的版本信息，MATLAB R2007a 中 Excel Link 的版本是 2.5）。然后单击【确认】按钮添加。稍后，MATLAB 会自动启动，Excel 的工具栏多了一行 Excel Link 的命令按钮。如图 10-26 所示。

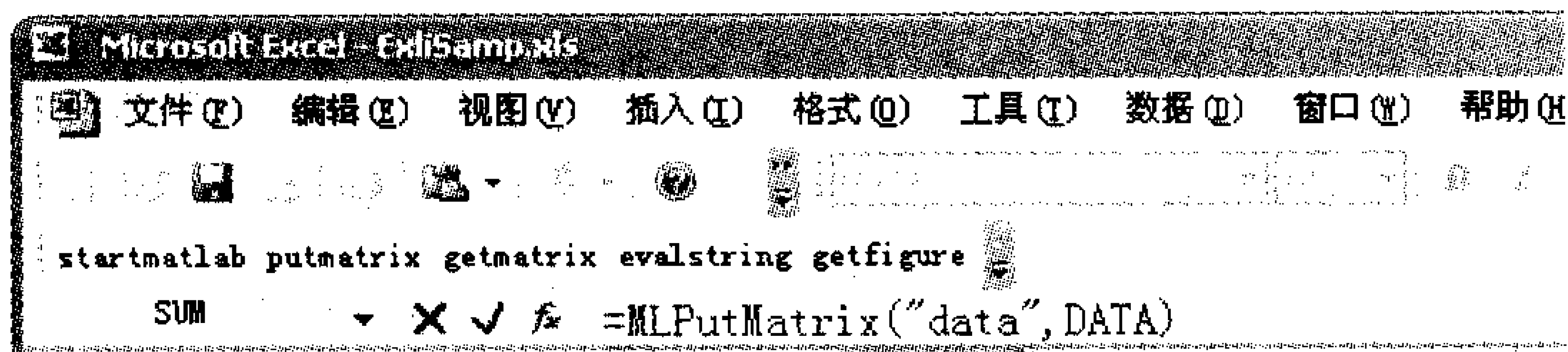


图 10-26 工具栏中的 Excel Link 按钮

我们可以发现多了 5 个命令按钮，分别为 startmatlab（启动 MATLAB）、putmatrix（将数据传送给 MATLAB）、getmatrix（从 MATLAB 获取数据）、evalstring（执行 MATLAB 的命令）和 getfigure（调用 MATLAB 绘图）。

此时就可以在 Excel 电子表中或宏工具使用 Excel Link 来实现 Excel 与 MATLAB 的交互了。在下次启动 Excel 时，会默认启动 Excel Link 和 MATLAB，关闭 Excel 时 MATLAB 也默认关闭。若不想在每次启动 Excel 时启动 MATLAB，则可以在电子表格中或函数输入框中输入以下命令：

```
=MLAutoStart("no")
```

当 Excel 中设置不自动启动 Excel Link 和 MATLAB 后，用户必须手动启动。有两种方法，第一种方法是 Excel 电子表格中调用 Excel Link 的 MLOpen 函数，即输入命令：

```
=MLOpen()
```

第二种方法是利用 Excel 的宏工具，在宏对话框中输入“MATLABinit”，然后单击【执行】按钮，同样可以启动 MATLAB。

当启动一个新的 Excel，可以与已经存在的 MATLAB 进程连接，而不必为此另外启动一个 MATLAB 进程。若让 Excel Link 启动时自动连接已有的 MATLAB 进程，则启动 MATLAB 时需加上“/automation”命令行选项，此选项表示启动 MATLAB 作为一个自动服务器，在自动服务器状态下，MATLAB 控制窗口最小化，桌面并不运行。方法为，用鼠标右键单击桌面上 MATLAB 的快捷方式，执行【属性】命令，然后单击“快捷方式”标签，在“目标”域添加“/automation”，注意，“matlab.exe”与“/automation”之间有个空格。

3. Excel Link 函数介绍

利用 Excel Link，Excel 就成为了 MATLAB 简单易用的数据存储与应用开发的应用终端，而 MATLAB 本身是一个数学计算和图像显示的强大工具。Excel Link 提供了两类函数：连接管理函数与数据管理函数来支持 Excel 与 MATLAB 的连接。以电子表格的输入公式或宏的方式即可调用这些函数，而不必脱离 Excel 环境。

Excel Link 提供了 4 个连接管理函数，具体功能如表 10-11 所示。

表 10-11　Excel Link 的连接管理函数

函　数	用　途
matlabinit	初始化 Excel Link，并启动 MATLAB 进程
MLAutoStart	自动启动 MATLAB 进程
MLClose	关闭 MATLAB 进程
MLOpen	启动 MATLAB 进程

其中 matlabinit 只可以从 Excel 的【工具】菜单中【宏】选项或【宏子函数】中使用，其他函数在 Excel 电子表格中和宏中都可以使用。

Excel Link 提供了如表 10-12 所示的数据管理函数，实现了在 Excel 与 MATLAB 直接传送数据和在 Excel 中执行 MATLAB 的命令等功能。

表 10-12 Excel Link 的数据管理函数

函数	用途
matlabfcn	调用 MATLAB 函数操作 Excel 中的数据
matlabsub	完成 matlabfcn 的功能，并指定结果值的输出位置
MLAppendMatrix	用 Excel 电子表格的数据创建或添加到 MATLAB 矩阵
MLDeleteMatrix	删除 MATLAB 矩阵
MLEvalString	在 MATLAB 中执行命令
MLGetFigure	将 MATLAB 当前图像导入到 Excel 电子表格中
MLGetMatrix	将 MATLAB 矩阵写到 Excel 电子表格中
MLGetVar	将 MATLAB 矩阵写入到 Excel 的 VBA 变量
MLPutMatrix	用 Excel 电子表格中数据创建或重写 MATLAB 矩阵
MLPutVar	用 Excel 的 VBA 变量创建或重写 MATLAB 矩阵
MLShowMatlabErrors	显示使用 MLEvalString 后返回的 Excel 和 MATLAB 的错误信息
MLStartDir	指定 MATLAB 启动后的工作目录
MLUseFullDesktop	指定 MATLAB 工作模式为桌面型或仅启动命令窗口

其中，MLPutVar 只能在宏中调用，其他的数据管理函数在电子表格和宏中都可以调用。使用 MLAppendMatrix、MLPutMatrix 以及 MLPutVar 将 Excel 中数据复制到 MATLAB；使用 MLEvalString 在 Excel 中执行 MATLAB 命令；使用 MLDeleteMatrix 删除 MATLAB 变量；使用 matlabfcn、matlabsub、MLGetMatrix 以及 MLGetVar 将 MATLAB 数据到 Excel。

4. 技巧与提醒

（1）Excel Link 的函数名不区分大小写，所以，函数 MLPutMatrix 和 mlputmatrix 一样。而 MATLAB 的函数名和变量名都区分大小写，所以 BONDS、Bonds 和 bonds 是 3 个不同的 MATLAB 变量。标准 MATLAB 函数名通常是小写的，如 plot(f)。

（2）在电子表格中输入公式，要以加号“+”或等号“=”开头，如下面代码：

```
=mlputmatrix("a", C10)
```

并且函数的参数用圆括号括起来。在宏中，函数名和第一个参数要间隔一个空格，不能使用圆括号。

电子表格中的函数成功执行后，此电子表格的内容会变成 0。

（3）若想自动重新计算函数值，可以将函数值加一个变化的值，如下面代码：

```
=MLPutMatrix("bonds", D1:G26) + C1
```

当 C1 中的数据变化时，Excel 会重新计算 MLPutMatrix 的函数值。需要注意的是，不要产生死循环，如函数执行会影响 C1 的值，而 C1 值的改变又触发函数的重新执行，这样就陷入了死循环。

（4）在 Excel 的电子表格中直接输入函数即可，不要使用 Excel 的函数向导，否则会产生不可预料的结果。

（5）若在宏中以子函数的形式调用 MLGetMatrix，则必须紧跟其后调用 MatlabRequest。

MatlabRequest 初始化 Excel Link 的变量，并且使 MLGetMatrix 可以作为子函数。如下面代码：

```
Sub Get_RangeA()
MLGetMatrix "A", "RangeA"
MatlabRequest
End Sub
```

（6）Excel Link 只处理 MATLAB 二维的数值数组和一维的字符数组（字符串）以及二维的细胞数组，而不能操作 MATLAB 的多维数组和结构体。

（7）Excel 的默认日期是从 1900 年 1 月 1 日开始。而 MATLAB 是从公元元年 1 月 1 日开始。所以 1996 年 5 月 15 日在 Excel 中是 35200，在 MATLAB 中是 729160，相差 693960。在 MATLAB 与 Excel 之间传送日期时，要加/减常数 693960 来调整。

5．编程实例

回归（Regression）技术和曲线拟合（Curve Fitting）试图找出描述变量间关系的函数，实际上建立了数据集的一个数学模型。MATLAB 提供了与此相关的运算符和函数，这些运算符简单易用，从而简化了整个工作。

例 10.37 使用 Excel 电子表格方式，结合 MATLAB 和 Excel 完成数据回归和曲线拟合。

本例用 Excel 的电子表格组织和显示数据，利用 Excel Link 函数将数据复制到 MATLAB，然后执行 MATLAB 的计算和绘图函数来实现数据回归和曲线拟合。

本例的源文件为 ExliSamp.xls，与 excllink.xla 在同一目录下，打开此文件后，选择 Sheet1 标签，如图 10-27 所示。

E5 ▾ fx =MLPutMatrix("data",DATA)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Regression and Curve Fitting												
2													
3		DATA			Excel Link Functions								
4	35	207	1325		1 Transfer the data to MATLAB.								
5	17	90	533		#MATLAB?	<== MLPutMatrix("data",DATA)							
6	43	180	1013										
7	41	187	1163		2. Set up data for regression								
8	177	552	5326		#MATLAB?	<== MLEvalString("y = data(:,3)")							
9	57	354	2043		#MATLAB?	<== MLEvalString("e = ones(length(data),1)")							
10	20	101	602		#MATLAB?	<== MLEvalString("A = [e data(:,1:2)]")							
11	18	91	532										
12	17	86	543		3. Compute regression coefficients.								
13	35	180	1134		#MATLAB?	<== MLEvalString("beta = A\y")							
14	25	136	766										
15	17	84	495		4 Calculate regressed result								
16	23	102	635		#MATLAB?	<== MLEvalString("fit = A*beta")							
17	24	148	913										
18	40	292	1591		5. Compare original data with regression results.								
19	25	126	671		#MATLAB?	<== MLEvalString("[y,k] = sort(y)")							
20	17	88	521		#MATLAB?	<== MLEvalString("fit = fit(k)")							
21	46	235	1319		#MATLAB?	<== MLEvalString("n = size(data,1)")							
22	37	204	1038										
23	15	68	458		6. Use MATLAB's polynomial solving functions for another curve fit.								
24	85	363	2904		#MATLAB?	<== MLEvalString("[p,S] = polyfit(1:n,y',5)")							
25	66	300	2006		#MATLAB?	<== MLEvalString("newfit = polyval(p,1:n,S)")							
26	39	161	938										
27	111	459	3282		7. Plot curves and add legend								
28	16	80	476		#MATLAB?	<== MLEvalString("plot(1:n,y,'bo',1:n,fit,'r:',1:n,newfit,'g'); legend('data','fit','newfit')")							

Sheet1 / Sheet2 / Sheet3 / Sheet4 / Sheet5 / Sheet6 /

图 10-27 ExliSamp.xls Sheet1 的内容

ExliSamp.xls 的 Sheet1 左面为数据区，并命名为 DATA，右面是 Excel Link 函数和注释。打开带有 Excel Link 函数的 Excel 文件时，会自动执行这些函数，按照从右向左、从下向上的顺序。由于此时没有打开 MATLAB 进程，所以执行错误。

首先应启动 MATLAB 进程，然后重新执行各个函数。首先选择 E5 单元格为活动单元，按【F2】键，然后按回车键执行 E5 单元格中的 Excel Link 函数“=MLPutMatrix("data", DATA)”，功能为将 DATA 数据传送到 MATLAB。DATA 数据集包含 3 个变量的 25 个观察值，从观察值中发现 3 个变量有很强的线性依赖关系。

按照同样的方法依次执行单元格 E8、E9、E10、E13、E16、E19、E20、E21、E24 和 E25 中的函数。若成功执行单元格的内容会变成 0。

单元格 E28 中的函数让 MATLAB 调用 plot 函数绘图，结果如图 10-28 所示。其中有原始数据（蓝圈）、回归结果 fit（红虚线）和 多多项式结果（绿实线）。

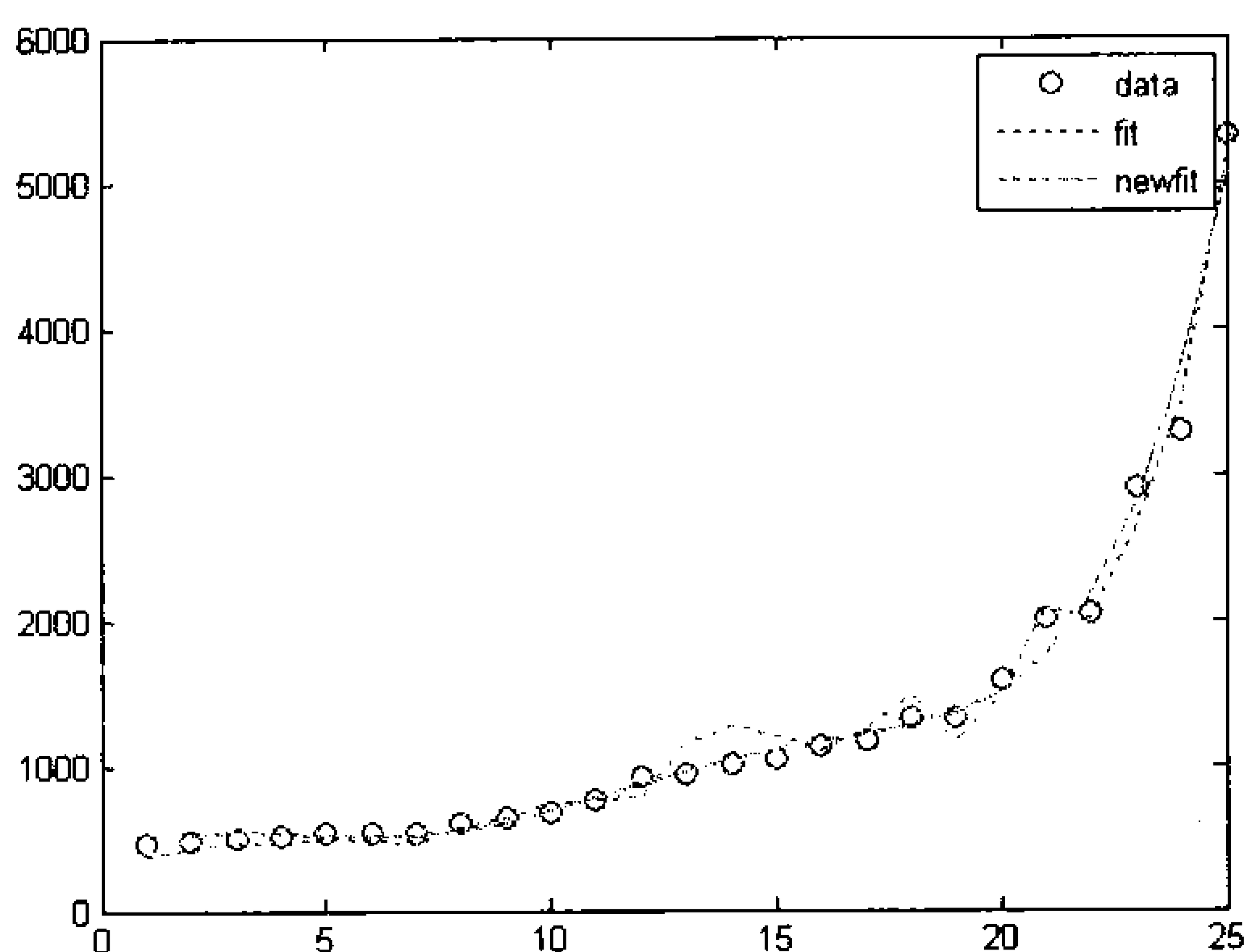


图 10-28　ExliSamp.xls 的绘图结果

由于数据很强的关系，但不是严格的线性依赖关系，fit 曲线（虚线）接近原数据但不完全一致。多项式曲线 newfit 为 data 数据更为精确的数学模型。

例 10.38　使用宏方法完成例 10.37。

打开 ExliSamp.xls 文件的 Sheet2 标签，利用 Visual Basic 的编辑器可以查看宏。如图 10-29 所示。

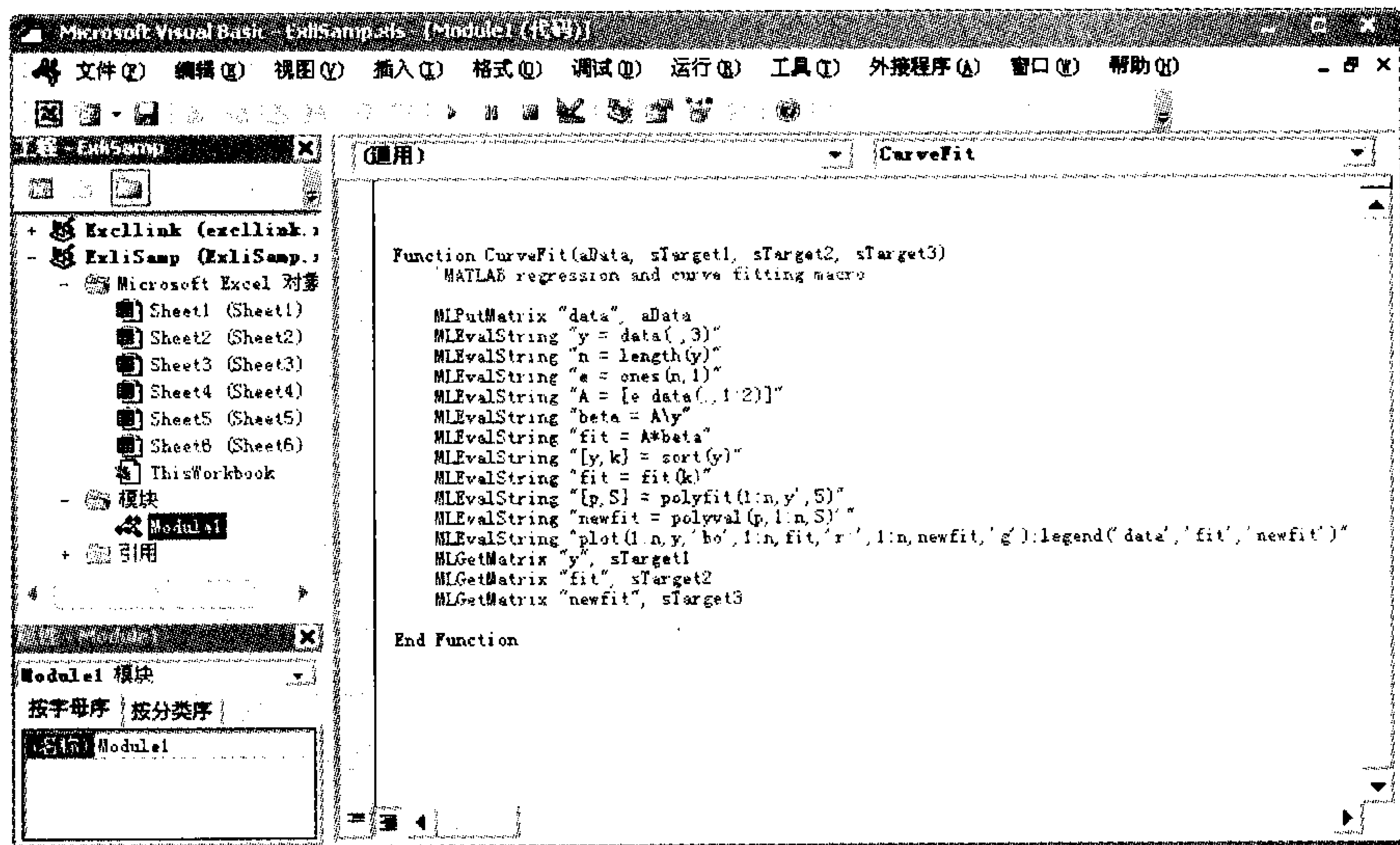


图 10-29　Visual Basic 编辑器界面

打开后，要确认 excllink.xla 已经成功加载。选择 Visual Basic 编辑器的【工具】菜单，执行【引用】命令，然后在对话框中看 excllink.xla 是否加载，若没有需要加载。

按照执行函数的方法执行电子表 Sheet2 的单元格 A4 中的 CurveFit 宏。CurveFit 宏执行了与电子表格例子相同的功能，包括结果绘图。另外，将数据 y、fit 和 newfit 复制到了电子表格中（宏中的最后 3 个 MLGetMatrix 函数完成了此功能）。

10.9.2　MATLAB Excel 生成器

MATLAB Excel 生成器（MATLAB Builder for Excel，也叫做 MATLAB Excel Builder）为 MATALB 编译器的一个扩展，能够将复杂的 MATLAB 算法转变成为 Excel 的插件，转变得到的文件可以在 Excel 表格中使用。无论是功能强大的 MATLAB 数学函数，还是复杂的图形函数算法，都可以被转变为 Excel 插件供用户使用。

Excel 生成器可将 MATLAB 的 M 函数转换为用户定义的类的方法，然后 Excel 生成器根据这个类创建组件。Excel 生成器创建的组件为 COM 对象，Excel 通过 VBA（Visual Basic for Applications）来访问。

MATLAB Excel 生成器有以下几个特点。

- 通过简单易用的图形界面完成 MATLAB 算法函数到 Excel 插件的转化。
- 自动生成 DLL 文件和相应的 VBA 文件，这些文件可被 Excel 电子表格直接使用。
- 创建生成的 DLL 插件文件运行速度远远超过直接使用 VBA 开发的算法。
- 隐藏数学算法，允许免费发布开发的 MATLAB 算法。
- 使用以矩阵为基本数据类型的 MATLAB 开发的算法能够大大提高程序的效率，可以加快程序的运行。
- Excel 生成器建立的 Excel 插件可以同一般的 Excel 插件很好的结合使用。
- Excel 生成器的图形用户界面方便易用，帮助用户生成 Excel 插件。图形用户界面包含了详尽的帮助信息和工程项目的管理，可以利用 Visual C++调试算法，还能够提供生成组件的详细过程。

1．创建可配置应用程序

1）用配置工具创建组件

使用 MATLAB 的配置工具或 mcc 命令可以创建组件。先介绍使用配置工具，其使用步骤类似于 MATLAB 编译器一节中配置工具的使用，同样主要包含创建新工程、添加文件、创建和打包 4 个步骤，具体参看 MATLAB 编译器的 10.6.4 节。

对于利用 Excel 生成器创建组件，有以下几个步骤。

（1）创建工程：在控制窗口输入 deploytool 命令打开配置工具，然后单击工具栏中的按钮创建工程，工程类型选择“MATLAB Builder for Excel”。

（2）添加文件：单击按钮添加预封装的文件，或者直接拖动文件到配置工具。

（3）设置属性（可选）：单击配置工具的工具栏中的按钮，打开属性设置窗口，对工程属性进行设置。

（4）保存工程。

（5）创建组件：单击工具栏中的按钮，启动创建组件。Excel生成器会将生成的中间文件复制到目录project_directory\src下，将必要的配置输出文件（一个DLL文件一个后缀为.bas的VBA文件）复制到project_directory\distrib目录下，project_directory为创建的工程根目录。

在\distrib目录下生成了.ctf文件（关于CTF文件，参见MATLAB编译器一节）。CTF文件使得应用程序在没有安装MATLAB的机器上可以正常运行。创建成功后，DLL文件会自动在系统中注册。

（6）测试组件，必要时重新创建。

2）用mcc命令创建组件

在MATLAB的控制窗口使用mcc命令同样可以创建组件，下面举例说明（关于mcc命令使用方法参见MATLAB编译器一节）。使用mcc命令时，目录\src和\distrib不会自动创建。若想创建这两个目录，并将相关文件复制其中，使用mcc命令的-d选项。

下面是mcc命令创建Excel生成器组件的基本语法：

```
mcc -W 'excel:<component_name>[,<class_name>[,<major>.<minor>]]'
```

其中component_name为预创建的组件名称，class_name为创建的类名，若不指定类，则采取组件名为类名。下面的例子调用mcc命令创建一个名为mycomponent的组件：

```
mcc -W 'excel:mycomponent,myclass,1.0' -T link:lib foo.m bar.m
```

此组件包含一个myclass类、两个方法foo和bar，版本号为1.0。

若为每个M文件生成Excel兼容的公式函数，则需使用-b选项，对于上例：

```
mcc -W 'excel:mycomponent,myclass,1.0' -b -T link:lib foo.m bar.m
```

另外可以采用cexcel束文件来简化命令行，下面例子不需要-T和-b选项：

```
mcc -B 'cexcel:mycomponent,myclass,1.0' foo.m bar.m
```

3）测试组件

创建组件后，可以在Excel中测试。方法为将VBA文件导入到Excel的Visual Basic编辑器中，然后在Excel的电子表格中调用其中的函数。

将VBA文件导入到Excel的Visual Basic编辑器中的方法如下。

（1）打开Excel，执行【工具】→【宏】→【Visual Basic编辑器】命令。

（2）在Visual Basic编辑器窗口中执行【文件】→【导入文件】命令，然后在目录project_directory\distrib中选择创建的VBA文件。

当创建的工程包含必要的初始化代码和每个用到的MATLAB函数相应的VBA公式函数时，会创建Visual Basic模块。

4）配置组件

创建并测试组件后，就可以根据Excel生成器产生的VBA代码创建一个Excel的插件（.xla），方法为保存电子表格为.xla文件到project_directory\distrib目录下。具体步骤如下。

（1）启动Excel，打开宏的Visual Basic编辑器。

（2）在打开的Visual Basic窗口中，执行【文件】→【导入文件】命令，然后在distrib目录下选择生成的VBA文件（后缀为.bas）。

（3）关闭Visual Basic编辑器。

（4）在 Excel 窗口中执行【文件】→【保存为】命令。

（5）将此 Excel 文件保存为“Microsoft Office Excel 加载宏(*.xla)”，目录为 project_directory\distrib。

然后，就可以在其他的 Excel 程序中使用创建后的插件了。

5）打包和发布组件

在成功地编译模块和创建 Excel 插件后，可以打包并向其他用户发布。在配置工具窗口单击工具栏中的 按钮，启动打包程序。成功打包后，Excel 生成器创建一个自解压的可执行程序，包含文件如表 10-13 所示。

表 10-13　自解压文件包含的文件

文　件	用　途
<componentname>.ctf	CTF 档案文件，与平台相关，需与用户所用平台相符合
<componentname_projectversion>.dll	编译后的组件
_install.bat	自解压文件运行的脚本
<componentname>_pkg.exe	自解压文件，用于在其他机器上发布
*.xla	project_directory\distrib 目录下所有的 Excel 插件文件

如名为 exsample 的工程，打包后，在 exsample/distrib 目录生成 exsample.ctf、exsample_1_0.dll、_install.bat 以及 exsample_pkg.exe。

2．Excel 生成器编程

每个 Excel 生成器组件都被创建成 COM 对象，用户可以在 Excel 中通过 VBA 来访问。本节讨论 VBA 编程实现 Excel 生成器组件与 Excel 的集成。本节需要读者有 VBA 编程基础和 Visual Basic 编程基础，否则读者可以跳过这一节。

将 Excel 生成器组件集成到 VBA 工程可以通过创建一个代码 module 来实现，然后在此 module 中调用函数或子程序，调用方法处理错误。下面具体介绍 Excel 生成器组件的 VBA 编程的步骤。本节的例子代码中默认组件名为 mycomponent，默认类名为 myclass。

1）函数与子程序的选择

VBA 提供了两种基本程序类型：函数和子程序。在电子表格中可以直接调用 VBA 函数。函数形式适用于原 MATLAB 函数有一个或多个输入并返回一个标量。

可以用宏的形式调用子程序。在原 MATLAB 函数返回一个数组或多个返回值时，宜采用子程序调用方式，因为需要将多个返回值与电子表格的单元格相匹配。

2）初始化 Excel 生成器库

在使用 Excel 生成器组件前，需要在当前 Excel 中初始化所需的库。每个 Excel 生成器组件只初始化一次即可。

初始化需要调用 MWUtil 类的成员函数 MWInitApplication，此类是库 MWComUtil 的一部分。下面是例子代码：

```
Dim MCLUtil As Object
Dim bModuleInitialized As Boolean

Private Sub InitModule()
```

```
    If Not bModuleInitialized Then
        On Error GoTo Handle_Error
        If MCLUtil Is Nothing Then
            Set MCLUtil = CreateObject("MWComUtil.MWUtil")
        End If
        Call MCLUtil.MWInitApplication(Application)
        bModuleInitialized = True
        Exit Sub
Handle_Error:
        bModuleInitialized = False
    End If
End Sub
```

读者可将上述代码作为模板来使用，并且最好在调用 Excel 生成器组件的每个函数时，都先调用 InitModule。

3）创建类的实例

在调用一个类的方法（编译后的 MATLAB 函数）前，需要创建这个类的一个实例，VBA 提供了两个方法来创建类的实例：CreateObject 函数和 New 操作符。

使用 CreatObject 函数的例子代码：

```
Function foo(x1 As Variant, x2 As Variant) As Variant
        Dim aClass As Object
        Set aClass = CreateObject("mycomponent.myclass.1_0")
        ' 参数 mycomponent.myclass.1_0 为类的标识符，即 ProgID
        ' 创建成功后，即可调用 aClass 类的方法
End Function
```

使用 New 操作法的例子代码：

```
Function foo(x1 As Variant, x2 As Variant) As Variant
    Dim aClass As mycomponent.myclass

    Set aClass = New mycomponent.myclass
    ' 调用 aClass 类的方法
End Function
```

上面两种方法是等效的，使用 New 操作法方法时，首先需要在 VBA 工程中引用包含此类的库。在 Visual Basic 编辑器窗口中执行【工具】→【引用】命令，在可用的库列表中，选择所需的库，比如上例需要选中 mycomponent 1.0 类型库。

4）调用类的方法

创建类实例后，就可调用类的方法访问编译的 MATLAB 函数。类的方法的参数列表有固定的模式。若方法有返回值，则第一个参数为 Long 型的 nargout，即返回值的个数；若没有返回值，则没有 nargout 参数。接下的参数为原 MATLAB 函数的输出参数和输入参数，顺序与原 MATLAB 函数的参数顺序一致。

例 10.39　演示 VBA 以函数形式调用 Excel 生成器组件中类的方法。

具体代码如下：

```
Function foo(x1 As Variant, x2 As Variant) As Variant
    Dim aClass As Object
    Dim y As Variant
```

```
        aClass = CreateObject("mycomponent.myclass.1_0")
        Call aClass.foo(1,y,x1,x2)
        foo = y
    End Function
```

其中 foo 方法有 4 个参数，1 为输出参数的个数，y 为输出参数，x1 和 x2 为两个输入参数，此方法对应与下面的 MATLAB 函数：

```
    function y = foo(x1,x2).
```

例 10.40 以子程序形式重写例 10.39 的代码，并采用 Excel 的 range 类型来处理输入和输出。

具体代码如下：

```
    Sub foo(Rout As Range, Rin1 As Range, Rin2 As Range)
        Dim aClass As Object

        aClass = CreateObject("mycomponent.myclass.1_0")
        Call aClass.foo(1,Rout,Rin1,Rin2)
    End Sub
```

5）处理 varargin 和 varargout 参数

当原 MATLAB 函数的参数中包含 varargin 或 varargout 时，这些参数将作为最后的输入/输出参数添加到参数列表中。可创建一个 Variant 数组，每个元素依次赋值为 varargin 或 varargout 的每个元素值。

例 10.41 创建一个 varargin 数组，以调用 MATLAB 函数：y = foo(varargin)

具体代码如下：

```
    Function foo(x1 As Variant, x2 As Variant, x3 As Varaint, x4 As Variant, x5 As Variant) As Variant
        Dim aClass As Object
        Dim v(1 To 5) As Variant
        Dim y As Variant

        v(1) = x1
        v(2) = x2
        v(3) = x3
        v(4) = x4
        v(5) = x5
        aClass = CreateObject("mycomponent.myclass.1_0")
        Call aClass.foo(1,y,v)
        foo = y
    End Function
```

MWComUtil 库中的 MWUtil 类提供了创建 varargin 参数的函数。

例 10.42 调用 MATLAB 的函数：varargout = foo(x1,x2)。

本例将 varargout 参数分为 3 个单独的 Excel 的 Range 参数，调用了库中的 MWUnpack 函数，具体代码如下：

```
    Sub foo(Rout1 As Range, Rout2 As Range, Rout3 As Range, Rin1 As Range, Rin2 As Range)
        Dim aClass As Object
        Dim aUtil As Object
        Dim v As Variant

        aUtil = CreateObject("MWComUtil.MWUtil")
```

```
        aClass = CreateObject("mycomponent.myclass.1_0")
        Call aClass.foo(3,v,Rin1,Rin2)
        Call aUtil.MWUnpack(v,0,True,Rout1,Rout2,Rout3)
    End Sub
```

6）处理错误

一个处理错误的简单方法是在可能出现错误的地方设置跳转语句，发生错误时跳转到处理错误行。结构如下：

```
    Sub foo( )
        '用户代码
        On Error Goto Handle_Error
        '用户代码
        Exit Sub
    Handle_Error:
        '处理错误
    End Sub
```

3．应用实例

例 10.43 利用 MATLAB 的 magic 函数在 Excel 电子表格中创建魔方矩阵。

此例子文件在 matlabroot\toolbox\matlabxl\examples\xlmagic 文件夹下，复制此文件夹到 MATLAB 当前目录。xlmagic 文件夹有两个文件，M 文件 mymagic.m 和 Excel 文件 xlmagic.xls。其中 mymagic.m 内容如下：

```
    function y = mymagic(x)
    %原注释略（笔者注）
    y = magic(x)
```

mymagic 函数仅调用了 MATLAB 函数 magic，以一个整数 N 为输入，生成一个 N 阶的魔方矩阵。Excel 文件 mymagic.xls 以 3 种不同方式调用了此函数。

- 调用 mymagic 函数生成 4 阶魔方矩阵，并在相应的 Excel 电子表格中输出。
- 将上面的矩阵转置。
- 生成更大阶数的魔方矩阵，同样在 Excel 电子表格中输出。

根据前面介绍的步骤创建“MATLAB Builder for Excel”工程，工程名设为 xlmagic，工程的类名为 xlmagicclass，为工程添加文件 mymagic.m。然后单击按钮创建组件。

至此，Excel 生成器完成了组件的创建，下面演示如何在 Excel 程序中使用此组件。

在打开例子文件前，需要查看 Excel 中关于宏的安全级别的设置。

打开 Excel，执行【工具】→【宏】→【安全性】命令，在“安全级”标签中选择“中”，此时每次打开带宏的 Excel 文件，会让用户选择是否启动宏。“安全级”也可选择“低”，此时打开带宏的 Excel 文件时，会自动加载宏。但是不推荐选择“低”，因为某些宏可能存在潜在的不安全因素。安全级别选择“中”后，再次打开 mymagic.xls，Excel 会询问是否启用宏，选择启用宏。

在 Excel 窗口按【Alt+F8】组合键或执行【工具】→【宏】→【宏】命令，在执行宏的对话框中选择“mymagic”，然后单击【执行】按钮。“mymagic”宏的作用是创建魔方矩阵。执行结果如图 10-30 所示。

	A	B	C	D	E
1					
2	4	16	2	3	13
3		5	11	10	8
4		9	7	6	12
5		4	14	15	1
6					

图 10-30　宏 mymagic 的执行结果

用 Visual Basic 编辑器查看宏 mymagic 的代码，主要内容如下（做了删减）：

```
Sub mymagic()
    Dim R1 As Range ' 输入
    Dim R2 As Range ' 输出

    Set R1 = Range("A2")        ' 将 Excel 单元格 A2 的值作为输入
    Set R2 = Range("B2:E5")  ' 将结果输出到 B2:E5 范围的单元格中

    Call xlmagicclass.mymagic(1, R2, R1)
    '调用 xlmagicclass 类的 mymagic 方法创建魔方矩阵
End Sub
```

其他两个宏 mymagic_transpose 与 mymagic_resize 的执行与原理类似，在此不再赘述。

10.10　MATLAB Java 生成器

Java 有许多值得称道的优点，如简单、面向对象、分布式、可靠、安全、结构中立性、可移植性、高性能、多线程和动态性等，用 Java 开发的程序可以在网络上传输，并运行于任何客户机上。利用 Java 生成器可以将 MATLAB 函数封装到 Java 程序中供 Java 调用，从而将 MATLAB 的计算与绘图优势与 Java 结合起来。

本章主要介绍创建和打包并发布 MATLAB Java 生成器组件，最后介绍了 Java 生成器编程，以及创建使用 Java 生成器组件的 Java 应用程序。

10.10.1　Java 生成器创建组件

和 MATLAB Excel 生成器一样，MATLAB Java 生成器（MATLAB Builder for Java，也称做 Java Builder）也是 MATLAB 编译器的一个扩展，也同样使用配置工具。

使用 Java 生成器可以将 MATLAB 的函数封装到一个或多个 Java 类中，作为 Java 组件或打包使用。每个 MATLAB 函数被封装成 Java 类的方法，在 Java 应用程序中可以调用。打包或发布此应用程序时，必须包含 Java 生成器产生的一些文件，以及 MATLAB 编译器提供的 MCR（包含了 MCR 后，发布的应用程序可以在没有安装 MATLAB 的机器上正常运行，具体参见 MATLAB 编译器一节）。

利用 Java 生成器将 MATLAB 的 M 函数封装到 Java 的类中，编译后在 Java 中运行，一般需要完成下面几个步骤：设置环境、封装 M 函数、创建组件和测试组件。下面以实例介绍这几个步骤。

例 10.44　用 Java 生成器将 MATLAB 的 magic 函数封装成组件以创建魔方矩阵。

创建的 Java 组件名为 magicsquare，组件包含 magic 类、一个.jar 文件、一个.ctf 文件以及其他配置应用程序需要的文件。类 magic 封装了 MATLAB 函数 makesqr（具体由 magic 实现）来计算魔方矩阵。本例使用配置工具，读者也可以使用 mcc 命令。

第一步：设置环境。

建议使用 MATLAB 发布的 JRE，在目录 matlabroot\sys\java\jre\win32（或 win64）\jre.ctg 下。使用 Java 时，必须按照以下步骤设置开发环境。改变或忽略某一步骤，都可能会产生错误或得到不可预料的结果。

（1）将例子复制到 MATLAB 当前目录。

①在 MATLAB 当前目录创建文件夹 javabuilder_examples。

②在 javabuilder_examples 下创建子目录 magic_square。

③将 matlabroot\toolbox\javabuilder\Examples\MagicSquareExample 文件夹复制到目录 magic_square 下，此文件夹包括 M 文件 makesqr.m 和 Java 演示程序 getmagic.java。

（2）用系统 cmd 程序，用命令切换到\javabuilder_ examples\magic_square 目录。

```
cd \javabuilder_examples\magic_square
```

（3）设置 JAVA_HOME 变量为安装 JDK（Java Developer's Kit，Java 开发包）的位置。

还是在 cmd 程序，仍在\javabuilder_examples\magic_square 目录，输入以下命令：

```
set JAVA_HOME=JDK_pathname
```

其中，JDKpathname 为安装 JDK 的目录，并且此 JDK 必须与 Sun JDK 的 1.5.0 版本兼容。JAVA_HOME 最好定义为环境变量。

（4）在 MATLAB 控制窗口，输入命令：

```
getenv JAVA_HOME
```

返回的结果应该与刚设置的 JAVA_HOME 的路径相同。

第二步：封装 M 文件到 Java 类。

在编译 Java 程序前，需启动配置工具将 M 文件添加到 Java 类中。

（1）启动配置工具，然后创建 MATLAB Builder for Java 工程，工程命名为 magicsquare，步骤与 Excel 生成器应用类似。工程目录为\javabuilder_examples\magic_square。

（2）类名修改为 magic，将目录\javabuilder_examples\magic_square\MagicDemoComp 下的 makesqr.m 文件添加到工程中。

（3）在配置工具中的面板中，确认“Generate Verbose Output”选项处于选中状态，然后保存此工程。

第三步：创建可配置的客户端组件。

在客户端应用程序中使用 Java 生成器组件 getmagic.java，需做以下操作。

（1）创建工程，创建 Java 组件。

在配置工具的工具栏中单击按钮，启动创建组件。创建过程中，在\src 和\distrib 子目录下产生了必要的文件。

（2）创建 Java 类后，进入\javabuilder_examples\magic_square\magicsquare 目录，然后检查新创建的 src 和 distrib 文件夹的内容。文件夹 src 包含生成的 Java 源文件，文件夹 distrib 包含发布应用程序所需的文件：一个 Java 档案文件（.jar）和 CTF 文件。

（3）在 Windows 系统的 cmd 程序中，输入以下命令：

```
%JAVA_HOME%/bin/javac –classpath
matlabroot\toolbox\javabuilder\jar\javabuilder.jar;.\magicsquare\distrib\magicsquare.jar  .\MagicDem
oJavaApp\getmagic.java
```

此命令的每行的结尾必须只能有一个空格。

第四步：测试组件。

最后需要对创建的组件进行测试。 运行 getmagic 程序，其输入参数为魔方矩阵的阶数，本例是 5。运行此程序，会创建一个 magic 类，然后调用类的 makesqr 方法。方法 makesqr 使用 MATLAB 的 magic 函数计算魔方矩阵。在 Windows 的 cmd 程序中输入以下命令运行 getmagic：

```
matlabroot\sys\java\jre\architecture\jre_directory\bin\java classpath .;
matlabroot\toolbox\javabuilder\jar\javabuilder.jar;MagicDemoJavaApp\magicsquare\distrib\magicsqu
-are.jar getmagic 5
```

此命令的每一行的结尾必须只能有一个空格。

若程序成功执行，会在屏幕上输出一个 5 阶魔方矩阵，内容如下：

```
Magic square of order 5
Extracting CTF archive.....
17    24     1     8    15
23     5     7    14    16
 4     6    13    20    22
10    12    19    21     3
11    18    25     2     9
```

10.10.2 Java 生成器编程

要使用 MATLAB 的 Java 生成器创建和打包的 Java 组件，需解压缩后安装组件。开发使用 Java 生成器组件的 Java 应用程序还需要特别的编程技术，因为 MATLAB 编程与 Java 编程存在诸多不同，如数据类型，输入/输出参数的不同等。下面介绍开发使用 Java 生成器组件的 Java 应用程序的主要编程技术。

1．导入 Java 类和 MATLAB 库

在使用 MATLAB 的 Java 生成器创建的组件前，必须完成以下操作：

（1）用 Java 的 import 函数导入 MATLAB 库，内容如下：

```
import com.mathworks.toolbox.javabuilder.*;
```

（2）导入 Java 生成器创建的组件类，内容如下：

```
import com.mathworks.componentname.classname;
```

2．创建类的对象

像其他的每个 Java 类一样，Java 生成器创建的类在使用之前都要进行实例化。创建类的对象需要用 Java 的 new 函数。如已经创建了名为 MyComponent 的组件，组件类为 MyClass，则创建 MyClass 类的对象的语句如下：

```
MyClass ClassInstance = new MyClass();
```

3．参数传递

当调用 Java 生成器组件的方法时，接受的输入参数必须是 MATLAB 的数组格式。可

以在主调函数中手动转换，也可以传递 Java 数据类型的参数，调用时会自动转换。

1）手动转换成 MATLAB 类型

若是手动转换到标准的 MATLAB 数据类型，需使用 Java 生成器的 MWArray 类，具体参考 com.mathworks.toolbox.javabuilder 包。

魔方矩阵的例子采用了手动转换，下面是程序中的一段代码，将 java.lang.Double 参数转换成了 M 函数可以直接使用的 MWNumericArray 类型。

```
MWNumericArray dims = null;
dims = new MWNumericArray(Double.valueOf(args[0]), MWClassID.DOUBLE);
result = theMagic.makesqr(1, dims);
```

2）自动转换 MATLAB 类型

当只有少数的参数传递时，一般传递原 Java 的数据类型或对象比较有效。这种情况下，调用机制会将 Java 数据类型自动转换为等价的 MATLAB 类型。如 Java 中 double 型和 java.lang.Double 类的对象都被转换成 MATLAB 的 double 型。

魔方矩阵例子中，应用程序 getmagic 调用 makesqr 方法时，创建了 MWNumericArray 对象，因而是采用了手动转换。对于此例也可以采用自动转换，代码如下：

```
result = M.makesqr(1, arg[0]);
```

这种情况下，以 arg[0]传递了 Java 的 double 参数，又如下面的代码：

```
result = theFourier.plotfft(3, data, new Double(interval));
```

在这个 Java 表达式中，第三个参数是 java.lang.Double 型。根据转换规则，java.lang.Double 会自动转换成 MATLAB 的 1×1 的 double 数组。

转换规则在调用 MWArray 类的构造函数同样应用。如下面代码，MWNumericArray 类的构造函数调用时用 Java 的 double 变量作为输入参数：

```
double Adata = 24;
MWNumericArray A = new MWnumericArray(Adata);
System.out.println("Array A is of type " + A.classID());
```

Java 生成器会将输入参数转换成 MWNumericArray 类的对象。运行上面代码后结果如下：

```
Array A is of type double
```

在自动转换语句中，也可以指定要转换的数据类型。如下面的代码在调用 MWArray 类的构造函数时候指定 A 要转换成 MATLAB 中 1×1 的 16 位整数数组：

```
double Adata = 24;
MWNumericArray A = new MWnumericArray(Adata, MWClassID.INT16);
System.out.println("Array A is of type " + A.classID());
```

运行此例后，输入结果如下：

```
Array A is of type int16
```

4．错误处理

在执行 M 函数或数据类型转换时发生的错误被看为 Java 异常（Exception），异常包括 M 函数中的代码错误和 MATLAB 运行时错误。

检查出的异常必须由 Java 的 throws 语句抛出。Java 生成器组件只支持 com.mathworks.toolbox.javabuilder.MWException 异常。这个异常类继承自 java.lang.Exception 类。所有 MATLAB 运行时错误和用户代码错误都被当做 MWException 异常。在 Java 的应用程序中，

M 函数的接口都要用 throws 语句抛出 MWException 异常，如 myprimes 函数的接口如下：

```
public Object[] myprimes(int nargout, Object n) throws MWException
    {
        // 函数体省略
    }
```

每个调用 myprimes 的方法都应捕获 MWException 异常并处理。如 getprimes 函数调用 myprimes 方法的代码如下：

```
public double[] getprimes(int n) throws MWException {
    myclass cls = null;
    Object[] y = null;
    try
    {
        cls = new myclass();
        y = cls.myprimes(1, new Double((double)n));
        return (double[])((MWArray)y[0]).getData();
    }
    finally
    {
        MWArray.disposeArray(y);
        if (cls != null)
        cls.dispose();
    }
}
```

10.10.3 打包和发布 Java 应用程序

向用户打包和发布用到 Java 生成器的组件的应用程序时，必须包含 Java 生成器生成的支持文件和 MATLAB 编译器提供的 MCR（参见 MATLAB 编译器一节）。在创建和打包魔方矩阵例子中的 Java 组件后，通过下面步骤发布应用程序（在每台要使用组件的机器上都要重复下面的步骤，若是在创建组件的机器上使用则第一步可以省略）。

（1）若在开发应用程序的机器上没有安装创建的组件，则首先要用创建的打包文件安装组件。

（2）设置环境变量，需要设置 JAVA_HOME、CLASSPATH 和库目录变量。

（3）使用 JAVA 的 import 命令将 MATLAB 库和自己的 Java 类导入到代码中。

对于 magicsquare 例子，需要下面语句：

```
import com.mathworks.toolbox.javabuilder.*;
import magicsquare.*;
```

（4）创建应用程序中要用到的 Java 生成器的类的对象。

如创建 magic 类的对象 theMagic，用下面的语句：

```
theMagic = new magic();
```

（5）调用 Java 生成器中的类的方法。

创建类的对象后，就可以像调用 Java 类的方法一样调用 Java 生成器的类的方法。在魔方矩阵例子中，用下面语句调用 makesqr 方法：

```
result = theMagic.makesqr(1, n);
```

其中 n 是 MWArray 类的对象，下面是 n 的声明：

```
n = new MWNumericArray(Double.valueOf(args[0], MWClassID.DOUBLE);
```

（6）处理必要的数据类型转换。

（7）创建和测试 Java 应用程序。

10.10.4 应用实例

例 10.45 将 MATLAB 的 plot 函数封装成组件，在 Java 程序调用此组件实现绘图功能。

此例的源文件在 matlabroot\toolbox\javabuilder\Examples\PlotExample 目录下，包含一个 M 文件 drawplot.m 和一个 Java 程序 createplot.java。

此例的目的是演示以下功能：

- 使用 MATLAB 的 Java 生成器将 MATLAB 函数(drawplot)转化成 Java 类(plotter)的一个方法，然后封装到 Java 组件（plotdemo）中。
- 在 Java 应用程序（createplot.java）中访问 Java 生成器组件。其中通过创建 plotter 类的对象以及使用 MWArray 类库处理数据类型转换。
- 编辑、编译和运行 createplot.java 应用程序。

此例需要按以下步骤完成。

（1）将 PlotExample 文件夹复制到 MATLAB 当前的工作目录中。

（2）设置必要的环境变量。

（3）编写 MATLAB 下的 drawplot 函数，内容如下：

```
function drawplot(x,y)
plot(x,y);
```

上述代码在 PlotExample\PlotDemoComp\drawplot.m 文件中。

（4）按照以下步骤创建 Java 组件。

①在 MATLAB 中调用 deploytool 命令启动配置工具。

②创建新工程，类型为 MATLAB Builder for Java 工程，具体设置如下：工程（组件）名：plotdemo；类名：plotter；Show verbose output 选项选中。

③添加 M 文件 drawplot.m 到工程。

④保存工程。

⑤创建组件。

（5）编写访问组件的代码。

例子文件为 PlotExample\PlotDemoJavaApp\createplot.java Java 文件。此 Java 程序根据方程 $y = x^2$ 绘制了抛物线。代码如下：

createplot.java

```
import com.mathworks.toolbox.javabuilder.*;    /* 必要的导入 */
import plotdemo.*;

class createplot
{
    public static void main(String[] args)
```

```
{
    MWNumericArray x = null;    /*  x   */
    MWNumericArray y = null;    /*  y   */
    plotter thePlot = null;         /* 类 Plotter 的实例 */
    int n = 20;                     /* 要绘制的点的个数 */

    try
    {
        /* 为 x、y 分配空间 */
        int[] dims = {1, n};
        x = MWNumericArray.newInstance(dims,
            MWClassID.DOUBLE, MWComplexity.REAL);
        y = MWNumericArray.newInstance(dims,
            MWClassID.DOUBLE, MWComplexity.REAL);

        /* 设置每个 x 点的值，根据 x 值设置 y 值：y = x^2 */
        for (int i = 1; i <= n; i++)
        {
            x.set(i, i);
            y.set(i, i*i);
        }

        thePlot = new plotter();  /* 创建 plotter 对象 */
        thePlot.drawplot(x, y);    /* 根据 x、y 值绘图 */
    }

    catch (Exception e)
    {
        System.out.println("Exception: " + e.toString());      /*异常处理*/
    }

    finally
    {
        /* 释放资源 */
        MWArray.disposeArray(x);
        MWArray.disposeArray(y);
        if (thePlot != null)
            thePlot.dispose();
    }
}
}
```

这个程序完成以下功能。

①创建了两个 double 数组存储 x 和 y，使用 MWNumericArray 表示绘图数据。

②创建 plotter 类的对象 thePlot，代码如下：

```
thePlot = new plotter();
```

③调用 drawplot 方法用 MATLAB 的 plot 函数绘制方程，代码如下：

```
thePlot.drawplot(x,y);
```

④使用 try-catch 模块处理异常。

（6）用 javac 命令编译 createplot 程序。

在 Windows 命令行中输入以下命令：

```
javac –classpath
.;matlabroot\toolbox\javabuilder\jar\javabuilder.jar;
.\distrib\plotdemo.jar createplot.java
```

输入上述命令时，确保 matlabroot 参数中没有空格。javabuilder.jar;和.\distrib\plotdemo.jar 之间也没有空格。

（7）运行此应用程序。

运行 createplot.class 文件，在 Windows 命令行输入以下命令：

```
java –classpath
.;matlabroot\toolbox\javabuilder\jar\javabuilder.jar;
  .\distrib\plotdemo.jar
  Createplot
```

程序 createplot 运行后，结果如图 10-31 所示。

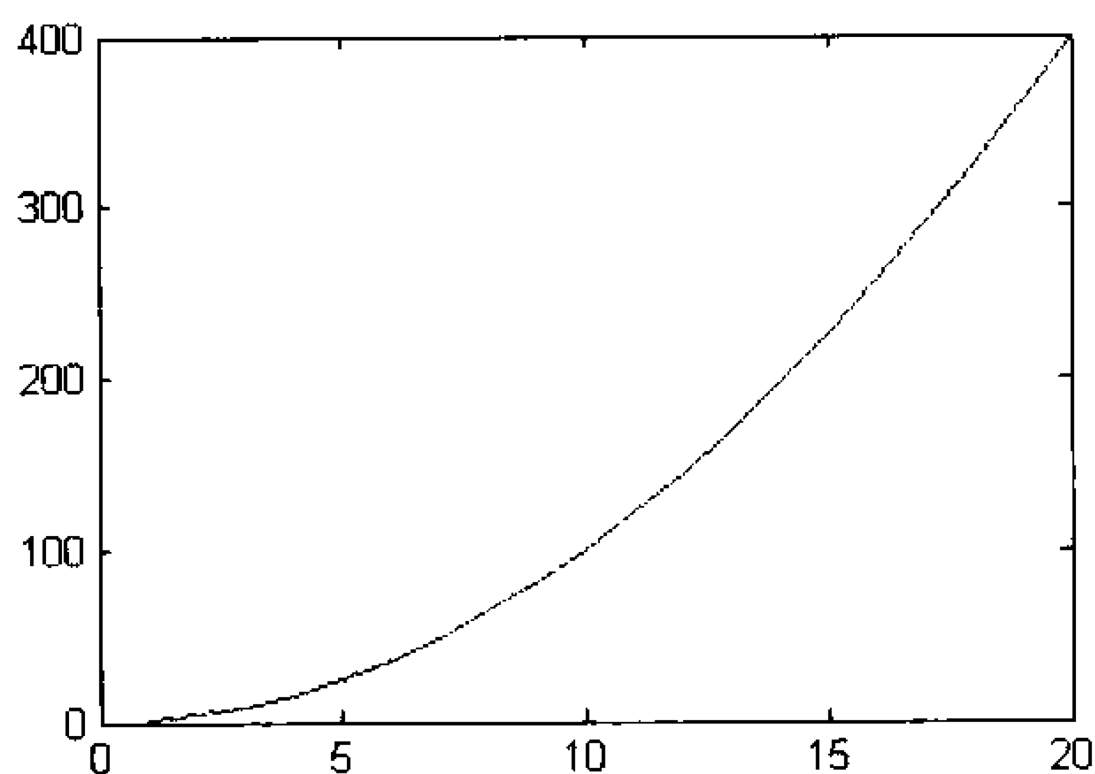


图 10-31　createplot 程序运行结果

10.11　MATLAB .NET 生成器

.NET 生成器同 MATLAB、MATLAB 工具箱、MATLAB 编译器一起，构建了一个集成化的算法和应用程序开发环境。通过 .NET 生成器创建的组件，.NET 程序员可以充分利用 MATLAB 的数学计算、图形绘制和数据分析等功能。.NET 组件同样适用于 Web 应用程序和服务器。

本章主要介绍用配置工具和 mcc 命令创建.NET 组件，另外，介绍了.NET 编程，最后以实例介绍了.NET 组件应用程序的开发。

10.11.1　.NET 生成器概述

与 Excel 生成器、Java 生成器一样，MATLAB 的 .NET 生成器（MATLAB Builder for .NET，也称做.NET Builder）也是 MATLAB 编译器的一个扩展，能够利用 MATLAB 算法代码自动生成独立的.NET assembly 类库或 COM 对象。可以通过任意符合公共语言规范（Common Language Specification，CLS）的编程语言，如 C#、VB.NET 等访问生成的.NET assembly 类库，通过任何符合 COM 规范的技术，如 Visual Basic、ASP 和 Microsoft Excel

等访问生成的 COM 对象。

CLS 支持.NET 公共语言运行库（Common Language Runtime），包括许多面向对象语言的特性，如 C#、VB.NET 以及 C++等。而不论是什么语言编写，只要符合 CLS，任意组件和工具之间都可以互操作。

.NET 生成器将 MATLAB 函数转换成.NET 方法。每个.NET 生成器组件都包含一个或多个类，每个类提供了与 M 函数的接口。为方便.NET 生成器编程，.NET 生成器提供了以下支持。

- 提供了符合 CLS 的方法打包 M 代码。
- 提供了数据类型转换、索引和数组格式化等功能，从而保持了 MATLAB 的灵活性。
- 为支持 MATLAB 的数据类型，.NET 生成器提供了 MWArray 数据转换类。
- 提供了错误处理，错误报告包含了出错 MATLAB 代码的引用，从而简化了调试。

10.11.2 创建 .NET 组件

创建.NET 组件同样有两种方法：图形化的配置工具和 mcc 命令，下面分别介绍。

1．用配置工具创建 .NET 组件

使用配置工具创建 .NET 组件比较简单，和利用配置工具创建独立运行程序或 C/C++库、Excel 组件、Java 组件过程类似，主要包括以下步骤。

（1）编写、测试，保存 .NET 组件要用到的 MATLAB 代码。

（2）启动配置工具窗口，创建新的“MATLAB Builder for .NET”工程，选择 .NET Component 类型。

（3）添加 MATLAB 函数文件。

（4）设置创建和打包的属性。

特别是在属性对话框的 .NET 页面设置.NET framework 的版本和预创建的 assembly 是 private 或 shared。

（5）保存工程。

（6）创建组件。

成功创建后，会在工程的\src 子目录下生成打包类的 C#代码 ，同时在\distrib 子目录下创建了组件和 CTF 文件。创建的 DLL 会自动在系统中注册。

（7）测试、必要时需编辑后重新创建。

在应用程序使用此组件之前和打包发布给其他用户使用之前都需要对此组件进行测试。

（8）创建可执行的自解压文件（可选）。

需要包含需要的 MATLAB 的 MCR，这样在没有安装 MATLAB 的机器上也可以使用此组件。这一步只有在需要打包发布组件给其他用户在不同机器上进行组件开发时才必要。

（9）在组件开发机器上发布、运行可执行的自解压文件。

2．用 mcc 命令创建 .NET 组件

用 mcc 命令同样可以创建 .NET 组件，创建.NET 组件的 mcc 语法如下所示，此命令

包含创建 .NET 组件所有的选项，包括必需的和可选的，方括号中的选项为可选的。

```
mcc –W 'dotnet:component_name,class_name, 0.0|1.1|2.0, Private|Encryption_Key_Path' file1
[file2...fileN][class{class_name:file1 [,file2,...,fileN]},... [-d output_dir_path] –T link:lib
```

下面分别介绍各个选项的作用。

- -W：表示创建打包函数，此选项后面的字符串参数意义如表 10-14 所示。

表 10-14　mcc 命令的-w 选项参数意义

-w 选项参数	意　义
dotnet:	表明是创建 .NET 组件，后面跟个冒号“:”
component_name	预创建的组件的名称
class_name	预创建的 .NET 类名
0.0|1.1|2.0	指定编译组件所用 .NET framework 的版本号，有 3 个值可选： 0.0 —— 目标机器上最新版本；1.1 —— 1.1 版本；2.0 —— 2.0 版本
Private|Encryption_Key_Path	设置创建的组件是私有或共享 assembly，若是共享，则需指定签署 assembly 的加密钥匙文件的完整路径

- file1 [file2...fileN]：设置要封装为类的方法的一个或多个 M 文件。
- class{class_name:file1 [,file2,...,fileN]},...：此选项可选，设置另外需要添加的类。设置此属性时在类名后加冒号，冒号后是此类的文件。
- [-d output_dir_path]：此选项可选，设置 .NET 生成器创建输出文件的目录。因为使用 mcc 命令时，工程目录下的子目录\src 和\distrib 不会自动创建。
- -T：设置输出类型，创建 .NET 组件时需设置为 link:lib，即最终创建 DLL 库文件。

可用 .NET 束文件来简化上面的 mcc 命令，但仍然需要传递-W 选项的 4 个字符串参数，不再需要-T 选项。

下面的 mcc 命令创建了一个 .NET 组件：

```
mcc -B 'dotnet:mycomponent,myclass,2.0,encryption_keyfile_path' foo.m bar.m
```

创建的组件名为 mycomponent，包含一个 .NET 类 myclass，这个类有 foo 和 bar 两个方法。其中 .NET 生成器使用 2.0 版本的.NET Framework 编译此组件。

使用 mcc 可以创建 .NET 组件的名字空间，如下面命令：

```
mcc - B 'dotnet:mycompany.mygroup.mycomponent,myclass,0.0,Private' foo.m bar.m
```

创建的 .NET 组件名为 mycomponent，名字空间为 mycompany.mygroup。在使用 myclass 类时的代码中，需先用下面的表达式：

```
using mycompany.mygroup;
```

使用 mcc 命令同样可以在.NET 组件中添加多个类，如下面命令：

```
mcc - B 'dotnet:mycomponent,myclass,2.0,Private' foo.m bar.m class{myclass2:foo2.m,bar2.m}
```

创建了名为 mycomponent 的 .NET 组件，此组件包含两个类：一个是 myclass 类，包含 foo 和 bar 两个方法；另一个是 myclass2 类，包含 foo2 和 bar2 两个方法。

10.11.3　.NET 生成器编程

创建好 .NET 组件并用 MATLAB .NET 生成器打包后，即可复制 packagename.exe 文件

到软件开发的机器运行来安装组件了。一般打包后的文件是可执行的自解压文件，运行后，会解压出相应的 CTF 文件、DLL 文件和 MATLAB MCR，并会在当前系统中自动注册组件。组件成功注册后，用户即可在本机上开发使用此组件的应用程序。

如同 Java 生成器编程一样，开发使用 .NET 组件的应用程序需要特别的技术支持，除了必须使用符合 CLS 的编程语言，如 C#、Visual Basic.NET 开发外，还主要包括以下几点。

1．设置组件 Assembly 和名字空间

在应用程序中使用组件 .NET 生成器生成的组件 assembly 之前，需要做以下工作。

（1）引用 MATLAB 数据类型转换 assembly 的名字空间，如：

```
using MathWorks.MATLAB.Arrays;
```

（2）引用.NET 生成器为用户组件生成的 assembly，如：

```
using MyComponentName;
```

.NET 生成器支持嵌套的名字空间。

例如，假设用户创建的组件名为 MyComponentName，若在名为 MyApp.cs 的 C#程序中使用此组件，需要下面的表达式：

```
using System;
using MathWorks.MATLAB.Arrays;
using MyComponentName;
```

2．创建类的对象

在程序中使用 .NET 类之前，需要创建 .NET 生成器生成的类的对象。例如，创建的组件的类为 MyComponentClass，可用下面语句创建这个类的对象：

```
MyComponentClass classInstance = new MyComponentClass();
```

3．数据类型转换

调用给予 MATLAB 函数的方法时，使用了 MATLAB 的一个数据类型转换类来传递参数和返回输出值。

下面例子代码用 MWNumericArray 类的构造函数由 int 型变量 data 显式地创建了一个数值常量 array。变量 array 可作为 .NET 生成器的方法的参数。

```
int data = 24;
MWNumericArray array = new MWNumericArray(data);
Console.WriteLine("Array is of type " + array.NumericType);
```

运行上面的例子，结果如下：

```
Array is of type double
```

此例中，原来的 int 型变量 data 转换成了包含一个 MATLAB double 型 1×1 数组的 MWNumericArray。而 doulb 型 1×1 数组正是 MATLAB 默认的数据类型，所以转换后的变量实现了 MATLAB 与 .NET 生成器的交互。

下面的几点介绍了更多的数据类型转换的细节。

1）指定转换类型

在数据类型转换时，默认的转换成 MATLAB 的 double 型数组，在 MWnumericArray 的构造函数中也可以指定要转换的类型。将可选的参数 makeDouble 设置为 False，则创建的 MATLAB 数组的类型为原变量的类型。

如下面的例子创建的 MATLAB 数组 array 的类型仍为变量 data 原来的类型：

```
short data = 24;
MWNumericArray array = new MWnumericArray(data, False);
Console.WriteLine("Array is of type " + array.NumericType);
```

运行上面的例子，得到以下结果：

```
Array is of type int16
```

2）处理不确定个数的参数

当 MATLAB 函数中的参数含有 varargin 或 varargout 时，则在不同的调用情况下，其输入/输出参数的个数有可能不同。.NET 生成器在封装这样的 M 函数时，提供了多个函数接口，保持了原 M 函数输入/输出参数可以任意个数的特点。

（1）含有 varargin 的 M 函数。

如下面的 M 函数 mysum：

```
function y = mysum(varargin)
y = sum([varargin{:}]);
```

函数 mysum 返回所有输入的和，而调用此函数时，可以提供任意多个的输入。对于 mysum 函数，.NET 生成器会生成下面的接口：

```
// 只有一个输出的接口
public MWArray mysum()
public MWArray mysum(params MWArray[] varargin)
// 标准接口
public MWArray[] mysum(int numArgsOut)
public MWArray[] mysum(int numArgsOut, params MWArray[] varargin)
// feval 接口
public void mysum(int numArgsOut, ref MWArray ArgsOut, params MWArray[] varargin)
```

原 M 函数的输入参数 varargin 用 MWArray[]数组传递，或以列表的形式将输入参数显式列出传递。利用重载 mysum 函数，使其保持了原 M 函数中输入个数可以任意多个的特点。下面的代码演示了在 .NET 程序中调用 mysum 方法的两种形式：

```
static void Main(string[] args]
{
MWArray sum= null;
MySumClass mySumClass = null;
try
    {
     mySumClass= new MySumClass();
     sum= mySumClass.mysum((double)2, 4);        // 两个输入参数
     Console.WriteLine("Sum= {0}", sum);
     sum= mySumClass.mysum((double)2, 4, 6, 8);  // 四个输入参数
     Console.WriteLine("Sum= {0}", sum);
    }
}
```

（2）含有 varargout 的 M 函数。

.NET 生成器利用类似处理 varargin 参数的方法处理 varargout 参数。如下面有 varargout 的 M 函数：

```
function varargout = randvectors()
for i=1:nargout
    varargout{i} = rand(1, i);
end
```

这个 M 函数返回了一个随机向量列表，第 i 个向量的长度等于 i。.NET 生成器为此 M 函数生成了以下 .NET 接口：

```
public void randvectors()
public MWArray[] randvectors(int numArgsOut)
public void randvectors(int numArgsOut, ref MWArray[] varargout)
```

3）处理返回值

前面的例子中都是知道返回值的数据类型和维数的，而在 MATLAB 编程中这些是未知的或者是可变的。这种情况下，调用此 M 函数的代码需要查询返回值的数据类型和维数。

有以下两种方法查询。

- 使用.NET 映像（Reflection）查询任意对象的数据类型。
- 使用 MWArray 类提供的方法查询 MATLAB 数组的信息。

下面分别详细介绍。

（1）使用.NET 映像查询。

下面的例子中，GetPrimes 函数调用了 myprimes 方法，然后使用映像查询其数据类型。本例假定返回值是数值类型，但是具体的类型未知。GetPrimes 函数主要代码如下：

```
public void GetPrimes(int n)
{
        MyPrimesClass myPrimesClass= new MyPrimesClass();
        MWArray primes= myPrimesClass.myprimes((double)n); //调用 myprimes(n)方法
        Array primesArray= ((MWNumericArray)primes).ToVector(MWArrayComponent.Real);

        if (primesArray is double[])     //判断 primes 的数据类型是否为 double
         {
              double[] doubleArray= (double[])primesArray;
              /* 返回值处理代码 ...*/
         }
        else if () // 然后用相同方法判断是否是 float[]、int[]、long[]、short[]、byte[]型，代码略
}
```

方法 toVector 是 MWNumericArray 类的一个方法，上例用 toVector 返回 .NET 的原数组 primesArray 来代表 MATLAB 基本数组类型，接下来挨个匹配查询。

（2）使用 MWArray 类查询。

对于上例，改用 MWNumericArray 类的 NumericType 方法结合 MWNumericType 枚举确定 MATLAB 基本数组的类型。主要代码如下：

```
public void GetPrimes(int n)
{
        MyPrimesClass myPrimesClass= new MyPrimesClass();
        MWArray primes= myPrimesClass.myprimes((double)n);
        if ((!primes.IsNumericArray) || (2 != primes.NumberofDimensions))
         {  //错误，返回}
        MWNumericArray _primes= (MWNumericArray)primes;
        MWNumericType numericType= _primes.NumericType;
        Array primesArray= _primes.ToVector(MWArrayComponent.Real);
        switch (numericType)
         {
```

```
            case MWNumericType.Double:
              {
                    double[] doubleArray= (double[])primesArray;
                    /* 处理 double 型的代码 ... */
                    break;
              }
            //下面同样用 case 语句查询是否是 Single、Int32、Int64、Int16、UInt8 类型，代码略。
            default:
              { //错误，返回}
        }
```

本例中同时调用 NumberOfDimensions 检查了返回值的维数。

4．错误处理

执行 M 函数或数据类型转换时可能会发生错误，即包括 MATLAB 运行时错误和用户代码错误。像一般 .NET 应用程序一样，调用 .NET 生成器的方法产生的错误有两种处理办法：本函数内处理异常和调用者处理异常。下面对这两种情况分别给出了例子。

（1）GetPrimes 函数内部处理异常。

```
public double[] GetPrimes(int n)
{
      try
        {
              MyPrimesClass myPrimesClass= new MyPrimesClass();
              MWArray primes primes= myPrimesClass.myprimes((double)n);
              return (double[])(MWNumericArray)primes).ToVector(MWArrayComponent.Real);
        }
      catch (Exception ex)
        {
              Console.WriteLine("Exception: {0}", ex);
              return new double[0];
        }
}
```

（2）用 myprimes 的函数处理异常。

```
public double[] GetPrimes(int n)
{
      try
        {
                    MyPrimesClass myPrimesClass= new MyPrimesClass();
                    MWArray primes= myPrimesClass.myprimes((double)n);
                    return(double[])(MWNumericArray)primes).ToVector(MWArrayComponent.Real);
        }
}
```

10.11.4 应用实例

本节用一个实例来说明.NET 组件的创建与使用.NET 组件的程序的开发。

例 10.46 将创建魔方矩阵的 M 函数封装为 .NET 组件，然后编写 Visual Basic.NET 程序使用这个组件。

此例所有源文件在 matlabroot\toolbox\dotnetbuilder\Examples\VS7\MagicSquareExample 目录下，其中组件所用的 M 文件 makesquare.m 在子目录\MagicDemo 下，Visual Basic.NET 的源程序 MagicDemoApp.vb 在子目录\MagicDemoVBApp 下。

按照 10.11.2 节中创建 .NET 生成器组件的步骤创建组件，组件名为 MagicDemoComp；类名为 MagicSquare。创建了 MagicDemoComp 组件后，就可以开发使用此组件的应用程序了，下面是 Visual Basic.NET 的源程序 MagicDemoApp.vb 的主要代码：

MagicDemoApp.vb

```
Imports System
Imports System.Reflection
Imports MathWorks.MATLAB.NET.Utility
Imports MathWorks.MATLAB.NET.Arrays
Imports MagicDemoComp
Namespace MathWorks.Demo.MagicSquareApp
Class MagicDemoApp
#Region " MAIN "
Shared Sub Main(ByVal args() As String)
    Dim arraySize As MWNumericArray = Nothing
    Dim magicSquare As MWNumericArray = Nothing
    Try
       ' 获取用户命令行的参数 或设置默认参数
       If (0 <> args.Length) Then
          arraySize = New MWNumericArray(System.Int32.Parse(args(0)), False)
       Else
          arraySize = New MWNumericArray(4, False)
       End If

       ' 创建魔方矩阵对象
       Dim magic As MagicSquare = New MagicSquare
       ' 计算魔方矩阵，并输出结果
       magicSquare = magic.makesquare(arraySize)
       Console.WriteLine("Magic square of order {0}{1}{2}{3}",
                          arraySize, Chr(10), Chr(10), magicSquare)
       ' 将魔方矩阵转换成.NET 程序中的二维数组，并输出结果
       Dim nativeArray(,) As Double = CType(magicSquare.ToArray(MWArrayComponent.Real),
                                      Double(,))
       Console.WriteLine("{0}Magic square as native array:{1}", Chr(10), Chr(10))
       ' 输出数组所以元素值：
       Dim index As Integer = arraySize.ToScalarInteger()

       For i As Integer = 0 To index - 1
          For j As Integer = 0 To index - 1
             Console.WriteLine("Element({0},{1})= {2}", i, j, nativeArray(i, j))
          Next j
       Next i
      Console.ReadLine() 'Wait for user to exit application
```

```
        Catch exception As Exception
        Console.WriteLine("Error: {0}", exception)

    End Try
    End Sub
    #End Region
    End Class
    End Namespace
```

上面的源程序完成了以下功能。

- 可以在命令行输入魔方矩阵的阶数。
- 将输入参数转换成了 MATLAB 整型标量。
- 声明了 MWNumericArray 型变量，以处理函数 makesquare 用到的数据。
- 创建了 MagicSquare 类的对象，名为 magic。
- 调用 magic 对象的 makesquare 方法，makesquare 方法使用 MATLAB 的 magic 函数生成魔方矩阵。
- 输出魔方矩阵的各个元素。

下面用 Visual Studio .NET 创建应用程序，新建 Visual Basic.NET 控制台程序，以文件 MagicDemoApp.vb 为源文件。或者也可使用本例的工程文件 MagicDemoVBApp.vbproj，此工程文件可在 Visual Basic.NET 中打开、编辑和编译。

对于此工程，还需要添加所用组件的引用（Reference），包括创建的 MagicDemoComp 组件和 MWArray 组件，其中，MagicDemoComp.dll 在 distrib 子目录下，mwarray.dll 在 matlabroot\dotnetbuilder\bin\win32\version_number 文件夹下，其中 version_number 为版本号。

最后在 Visual Basic.NET 中创建、运行应用程序查看运行结果。

第一次运行此程序时，Visual Basic.NET 在 MagicSquareExample\bin\debug 下创建名为 MagicDemo_MCR 的子目录，在 MagicDemo_MCR 包含 MagicSquare 组件封装的 M 函数的加密版本。